reinforcement learning through the optimization lens

Benjamin Recht
University of California, Berkeley

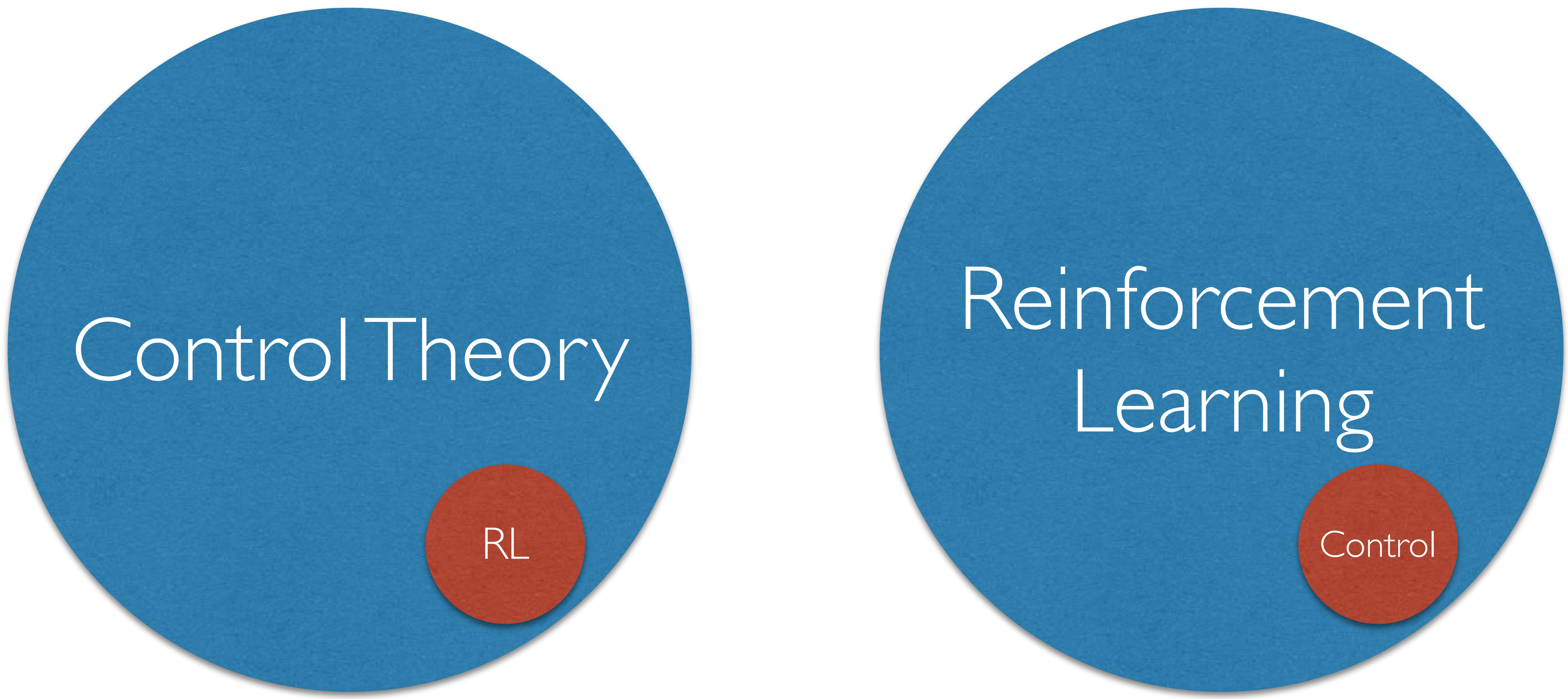
trustable, scalable, predictable

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 **CS**



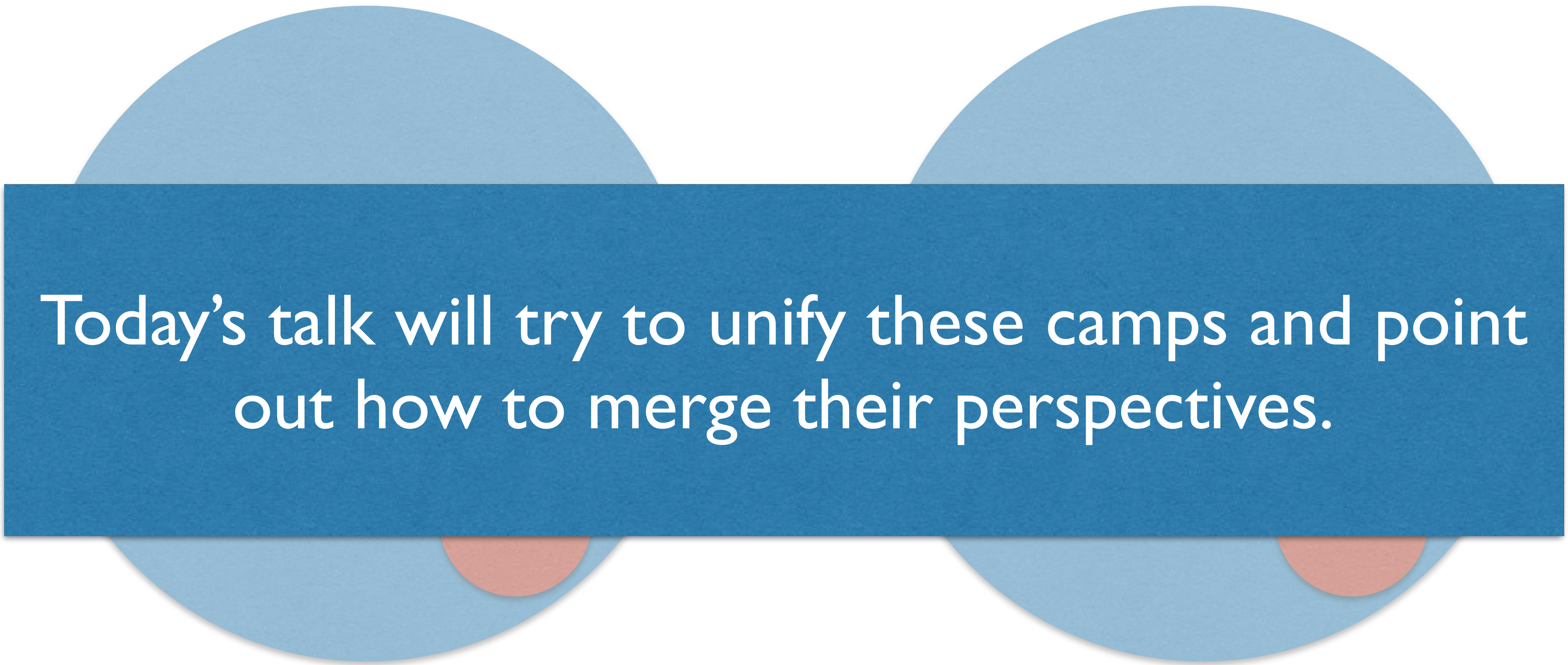
continuous *discrete*

model ⟶ *action* *data* ⟶ *action*

IEEE Transactions *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.

AE/CE/EE/ME	CS
continuous	*discrete*
model ⟶ *action*	*data* ⟶ *action*
IEEE Transactions	*Science Magazine*



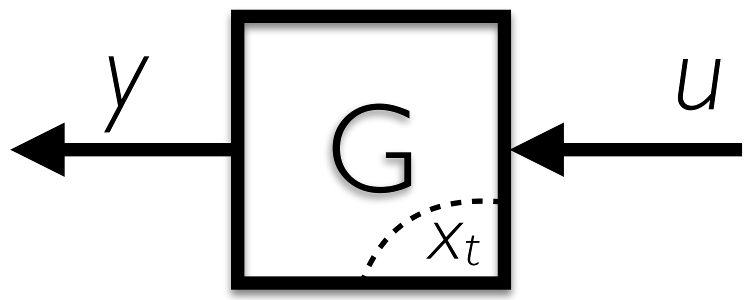
Main research challenge: What are the fundamental limits of learning systems that interact with the physical environment?

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



$$x_{t+1} = Ax_t + Bu_t$$
$$y_t = Cx_t + Du_t$$

Simplest case of such systems are *linear systems*

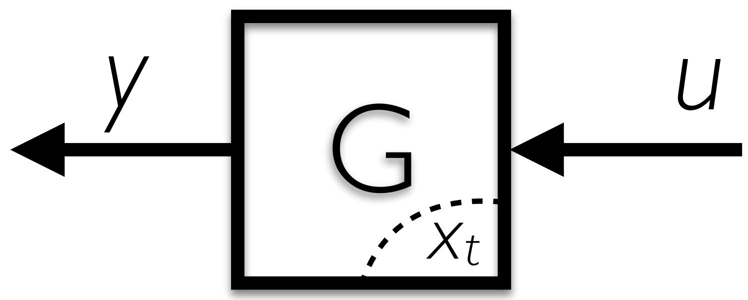
x_t is called the *state*, and the dimension of the state is called the *degree*, *d*.

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

y_t is called the *output*, and the dimension is q.

For today, will only consider *C=I*, *D=0* (x_t observed)

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



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

Simplest example: *Partially Observed Markov Decision Process (POMDP)*

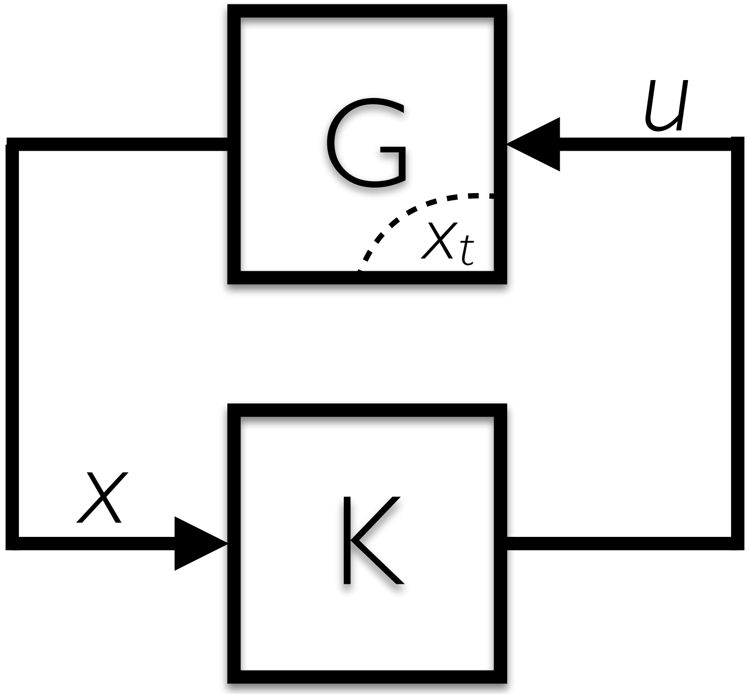
x_t is the *state*, and it takes values in [d]

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

y_t is called the *output*, and takes values in [q].

For today, will only consider when x_t observed (MDP).

Controller Design



$$x_{t+1} = Ax_t + Bu_t$$
$$u_t = \pi_t(\tau_t)$$

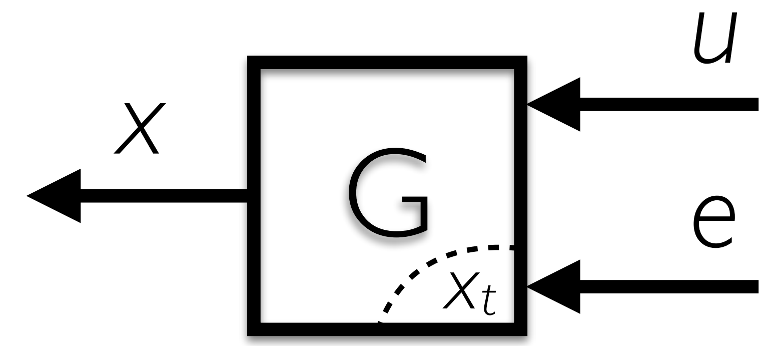
- A dynamical system is connected in feedback with a *controller* that tries to get the closed loop to behave.
- Actions decided based on observed *trajectories*

$$\tau_t = (u_1, \ldots, u_{t-1}, x_0, \ldots, x_t)$$

- A mapping from trajectory to action is called a *policy*, $\pi_t(\tau_t)$
- Optimal control: find policy that minimizes some objective.

Optimal control

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



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

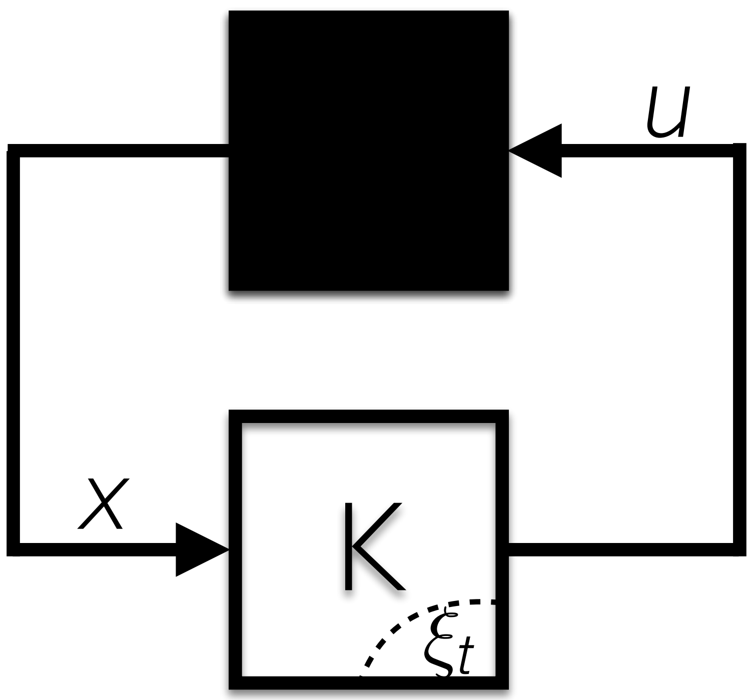
e_t is a noise process

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 <u>optimization decision variable</u>.

Optimal control

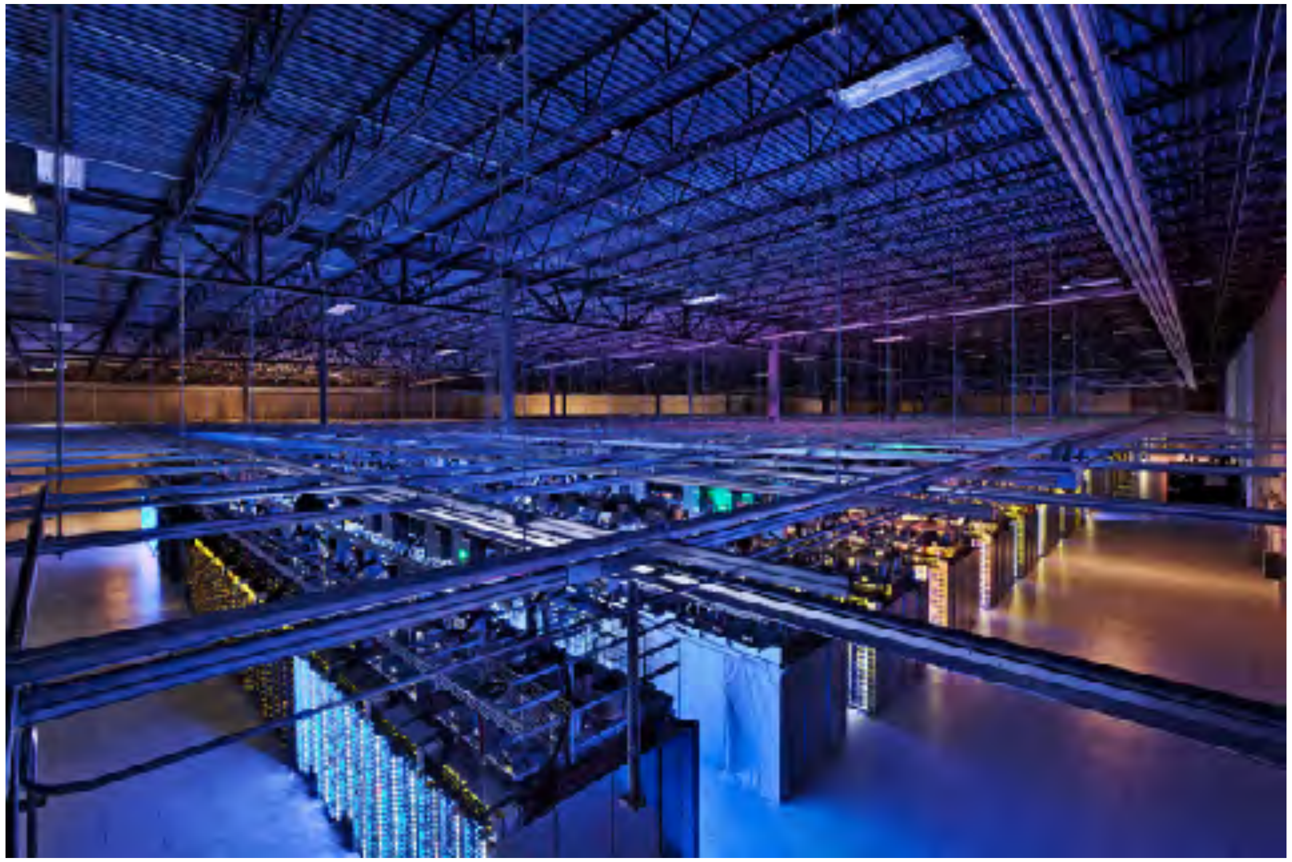


$$x_{t+1} = F(u_t, u_{t-1}, u_{t-2}, \ldots)$$
$$u_t = \pi_t(\tau_t)$$

- A dynamical system is connected in feedback with a *controller* that tries to get the closed loop to behave.
- Optimal control: find policy that minimizes some objective.

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

Today: Reinvent RL attempting to answer this question



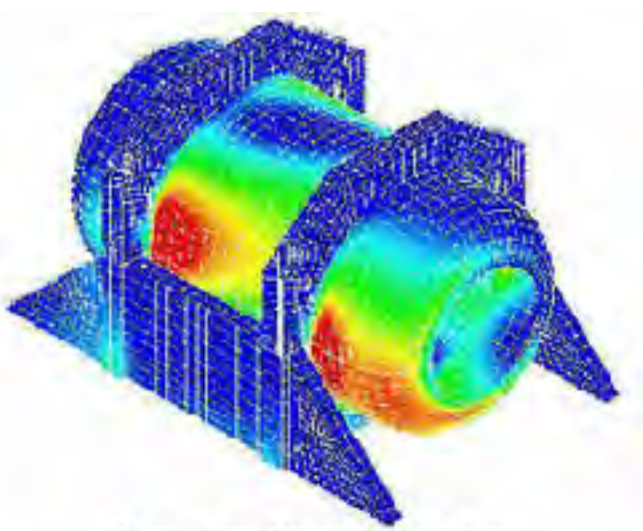
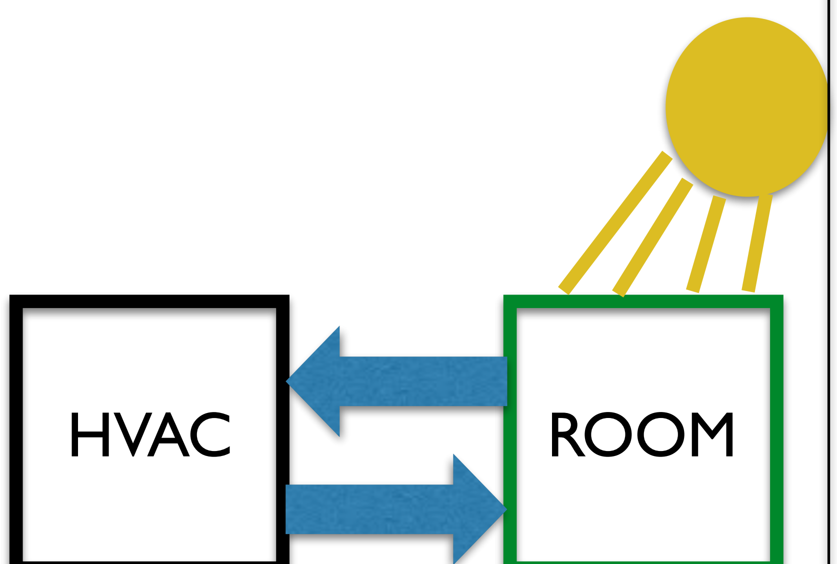
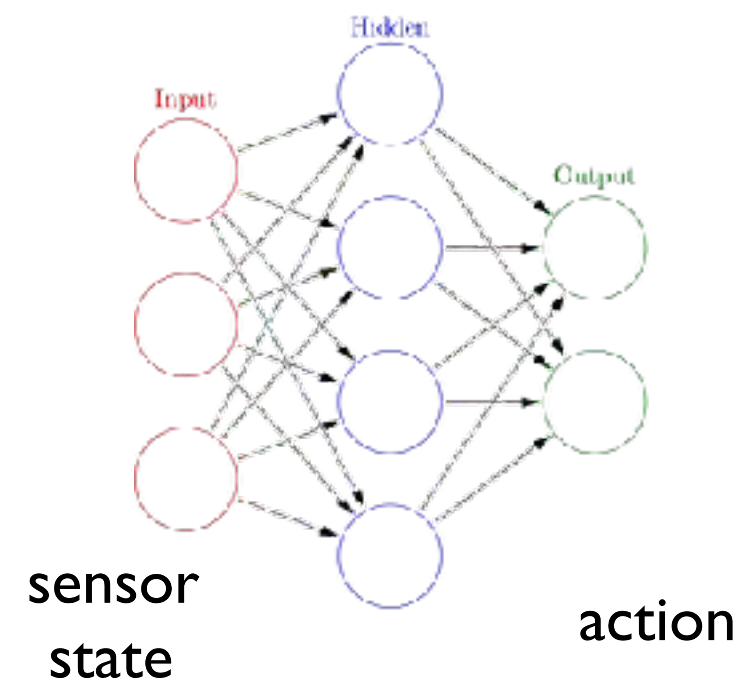


Figure 1 http://www.noraneng.com

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



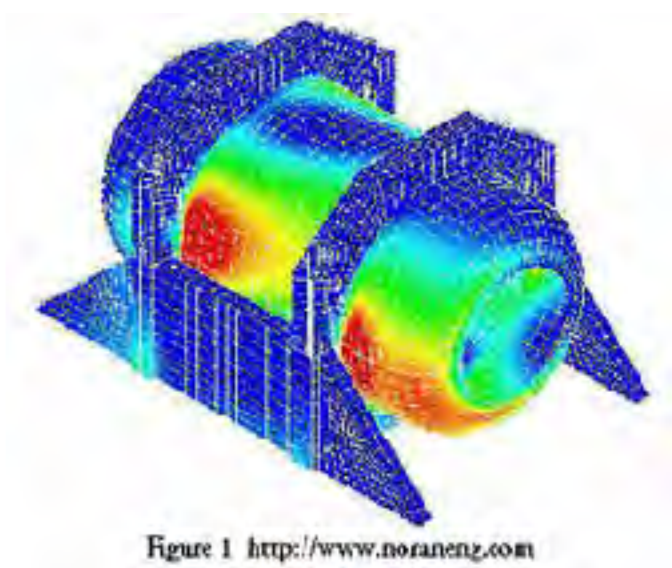
$$M\dot{T} = \dot{Q} + \dot{m}_s c_p (T_s - T)$$



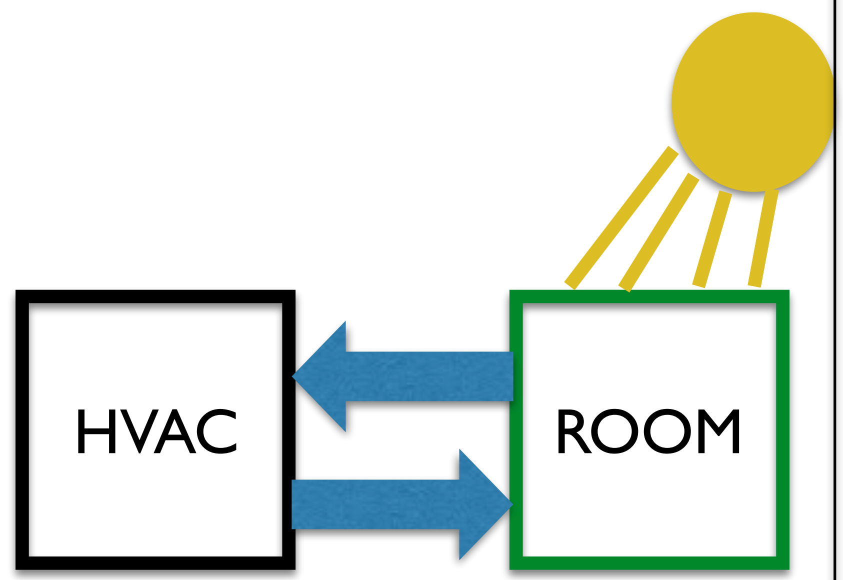
Identify everything

Identify a coarse model

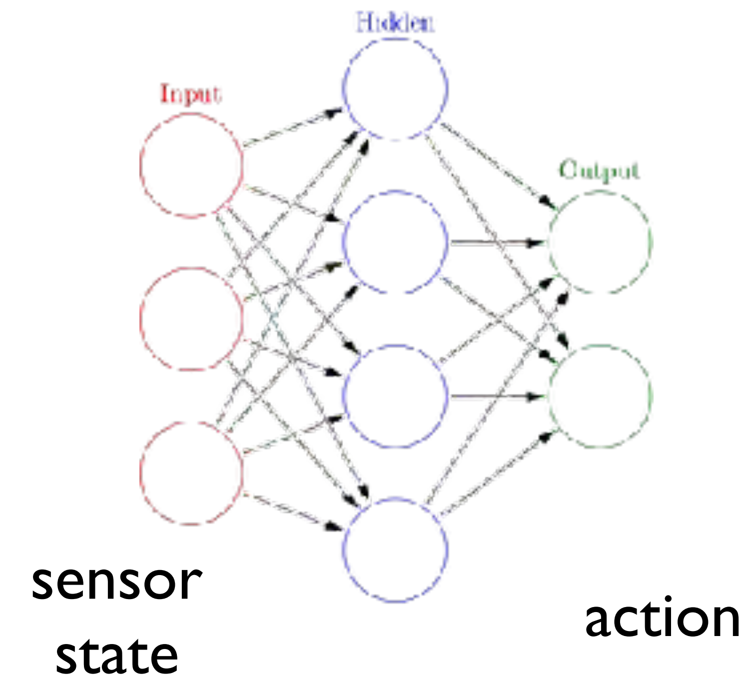
We don't need no stinking models!



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



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



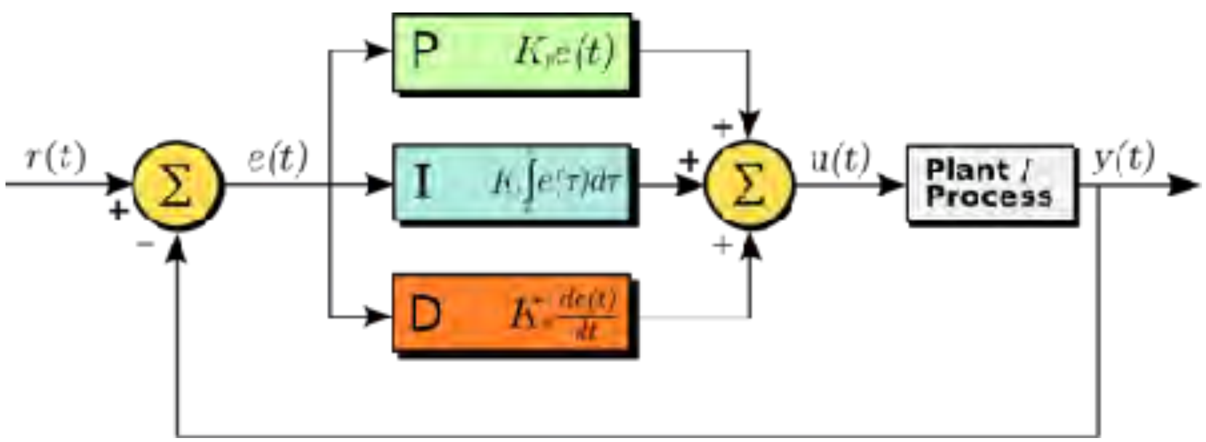
- PDE control
- High performance aerodynamics

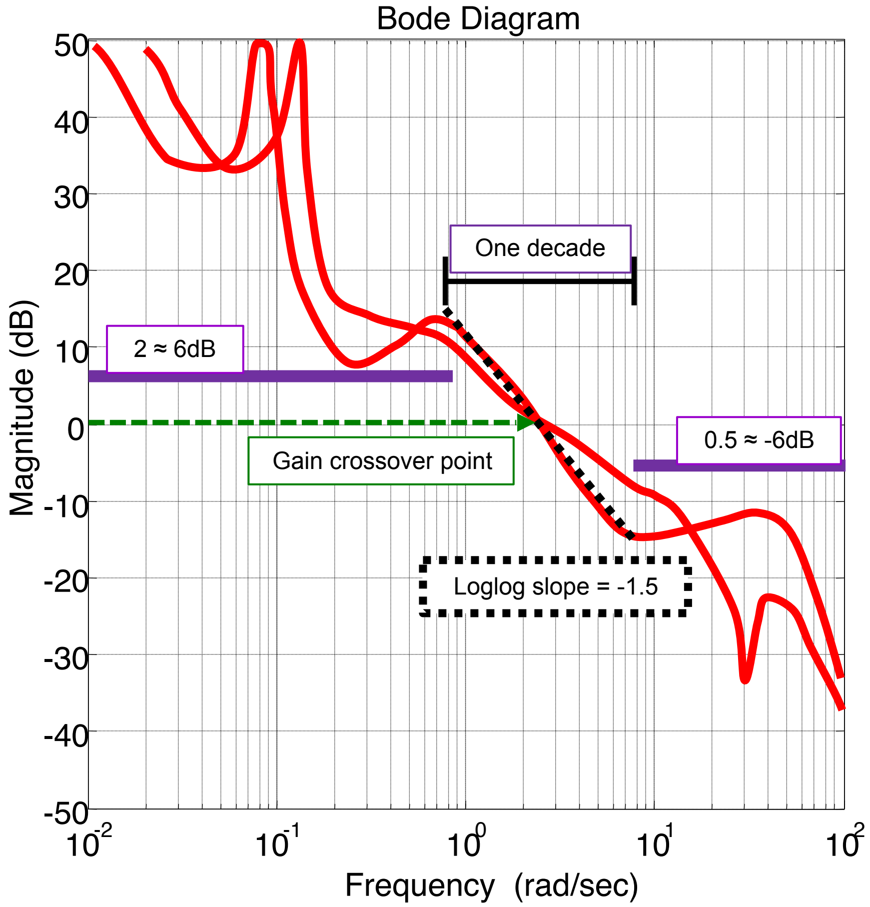
- model predictive control

- reinforcement learning
- PID control?

We need robust fundamentals to distinguish these approaches

But PID control works…





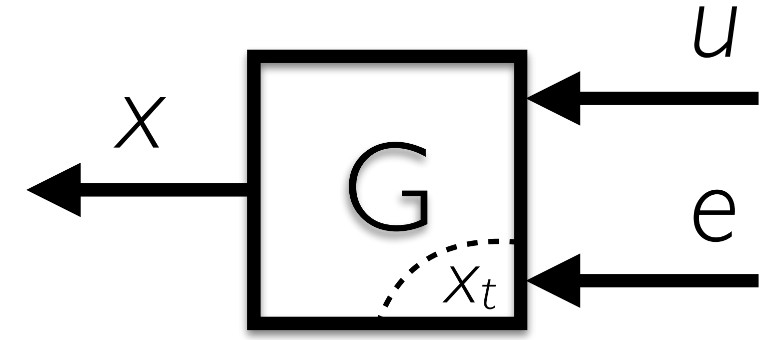
2 *parameters* suffice for 95% of all control applications.

How much needs to be modeled for more advanced control?

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

Optimal control

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



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 <u>optimization decision variable</u>.

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

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

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

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

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

"Simplest" Example: LQR

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

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

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

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

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

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

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

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

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

"Simplest" Example: LQR

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

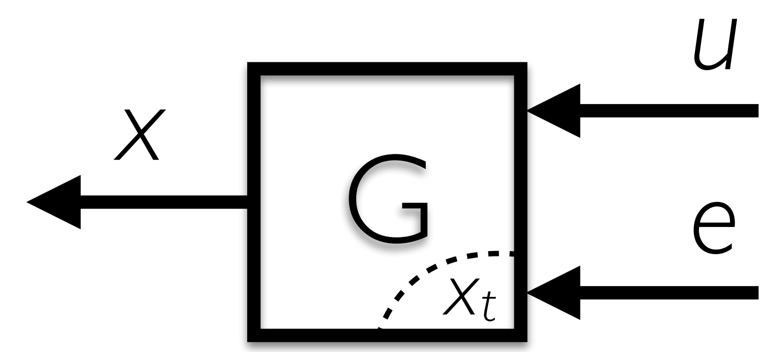
Suppose (A,B) unknown

What is the optimal estimation/design scheme?

How many samples are needed for near optimal control?

Optimal control

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

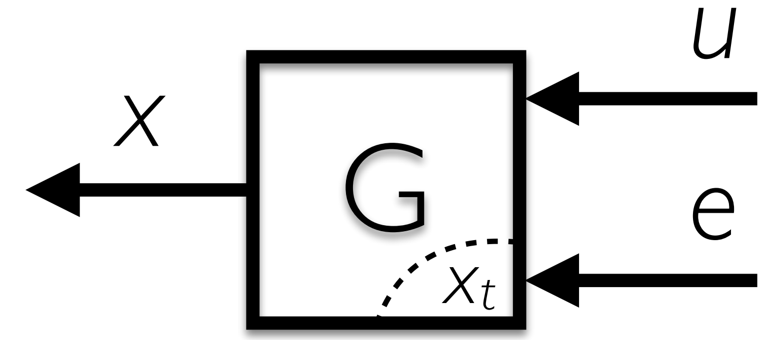


generic solutions with known dynamics:

Batch Optimization

Dynamic Programming

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

Model-based RL

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

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

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

Solve approximate problem:

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

"Simple" Example: LQR

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

s.t. $\quad x_{t+1} = Ax_t + Bu_t + e_t$ *Gaussian noise*

"Obvious strategy": Estimate (A,B), build control.

Run an experiment for T steps with random input. Then

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]

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

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

$\quad u_t = \pi_t(\tau_t)$

~~Approximate~~ Dynamic Programming

Dynamic Programming

$$\begin{aligned} \text{minimize} \quad & \mathbb{E}_e\left[\sum_{t=1}^{T} C_t(x_t, u_t) + C_f(x_{T+1})\right] =: V_1(x) \\ \text{s.t.} \quad & x_{t+1} = f_t(x_t, u_t, e_t), \;\; x_1 = x \\ & u_t = \pi_t(\tau_t) \end{aligned}$$

"Value function"

Terminal value:

$$\boxed{V_{T+1}(x) = C_f(x_{T+1})}$$

Recursive formula (recurse backwards):

$$\boxed{V_k(x) = \min_u C_k(x, u) + \mathbb{E}_e\left[V_{k+1}(f_k(x, u, e))\right]}$$

"Cost-to-go"

Optimal Policy:

$$\boxed{\pi_k(\tau_k) = \arg\min_u C_k(x_k, u) + \mathbb{E}_e\left[V_{k+1}(f_k(x_k, u, e))\right]}$$

"Simplest" Example: LQR

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

$\boxed{\textbf{Dynamic Programming:}}$ $V_T(x_T) = x_T^* P_T x_T$

$$V_{T-1}(x_{T-1}) = \min_{u_{T-1}} \mathbb{E}\left[x_{T-1}^* Q x_{T-1} + u_{T-1}^* R u_{T-1} + V_T(x_T)\right]$$

$$= \min_{u_{T-1}} \begin{bmatrix} x_{T-1} \\ u_{T-1} \end{bmatrix}^* \left\{ \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} + \begin{bmatrix} A & B \end{bmatrix}^* P_T \begin{bmatrix} A & B \end{bmatrix} \right\} \begin{bmatrix} x_{T-1} \\ u_{T-1} \end{bmatrix} + \sigma^2 \operatorname{Tr}(P_T)$$

$$\boxed{\begin{aligned} u_{T-1} &= -(B^* P_T B + R)^{-1} B^* P_T A x_T =: K_{T-1} x_{T-1} \\ V_T(x_{T-1}) &= x_{T-1}^* P_{T-1} x_{T-1} \\ P_{T-1} &= Q + A^* P_T A - A^* P_T B \left(R + B^* P_T B\right)^{-1} B^* P_T A \end{aligned}}$$

"Simplest" Example: LQR

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

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

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

Recursive formula:

$$\boxed{V_k(x) = \min_u C_k(x, u) + \mathbb{E}_e \left[V_{k+1}(f_k(x, u, e)) \right]}$$

Optimal Policy:

$$\boxed{\pi_k(\tau_k) = \arg\min_u C_k(x_k, u) + \mathbb{E}_e \left[V_{k+1}(f_k(x_k, u, e)) \right]}$$

Approximate Dynamic Programming

$$\begin{array}{ll} \text{minimize} & \mathbb{E}_e \left[\sum_{t=1}^{\infty} \gamma^t C(x_t, u_t)\right] \\ \text{s.t.} & x_{t+1} = f(x_t, u_t, e_t) \\ & u_t = \pi(\tau_t) \end{array}$$

discount factor

Bellman Equation:

$$\boxed{V_\gamma(x) = \min_u C(x,u) + \gamma \mathbb{E}_e \left[V_\gamma(f(x,u,e))\right]}$$

Optimal Policy:

$$\boxed{\pi(x) = \arg\min_u C(x,u) + \gamma \mathbb{E}_e \left[V_\gamma(f(x,u,e))\right]}$$

Generate algorithms using the insight:

$$V_\gamma(x_k) \approx C(x_k, u_k) + \gamma V_\gamma(x_{k+1}) + \nu_k$$

Approximate Dynamic Programming

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

$$\boxed{Q(x,u) = C(x,u) + \mathbb{E}_e \left[\gamma V_\gamma(f(x,u,e)) \right]}$$

Bellman Equation:

$$\boxed{V_\gamma(x) = \min_u C(x,u) + \gamma \mathbb{E}_e \left[V_\gamma(f(x,u,e)) \right]}$$

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

Optimal Policy:

$$\boxed{\pi(x) = \arg\min_u C(x,u) + \gamma \mathbb{E}_e \left[V_\gamma(f(x,u,e)) \right]}$$

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

Approximate Dynamic Programming

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

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

$u_t = \pi(\tau_t)$

$$\boxed{Q(x,u) = C(x,u) + \mathbb{E}_e\left[\gamma V_\gamma(f(x,u,e))\right]}$$

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

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

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

Q-learning:

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

Direct Policy Search

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

Sampling to Search

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

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

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

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

Reinforce Algorithm

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

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

Reinforce Algorithm

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

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

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

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

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

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

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

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?

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

Direct Policy Search

Policy Gradient

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

probabilistic policy

Random Search

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

parameter perturbation

Reinforce applied to either problems does not depend on the dynamics

Both are Derivative-free algorithms!

Direct Policy Search

Policy Gradient

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

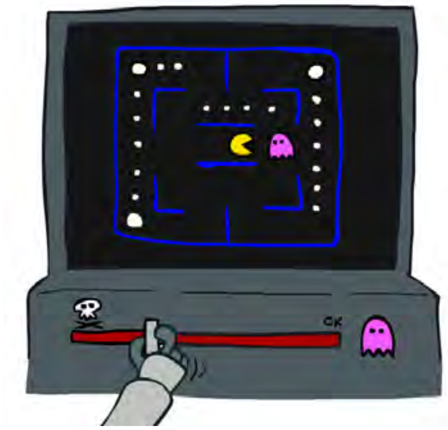
s.t. $\quad x_{t+1} = f_t(x_t,u_t,e_t)$

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

probabilistic policy

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

Policy Gradient for LQR



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

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

"Greedy strategy": Build control $u_t = Kx_t$

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

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

Direct Policy Search

Random Search

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

parameter perturbation

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

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

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

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

random finite difference approximation to the gradient

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

Random Search for LQR

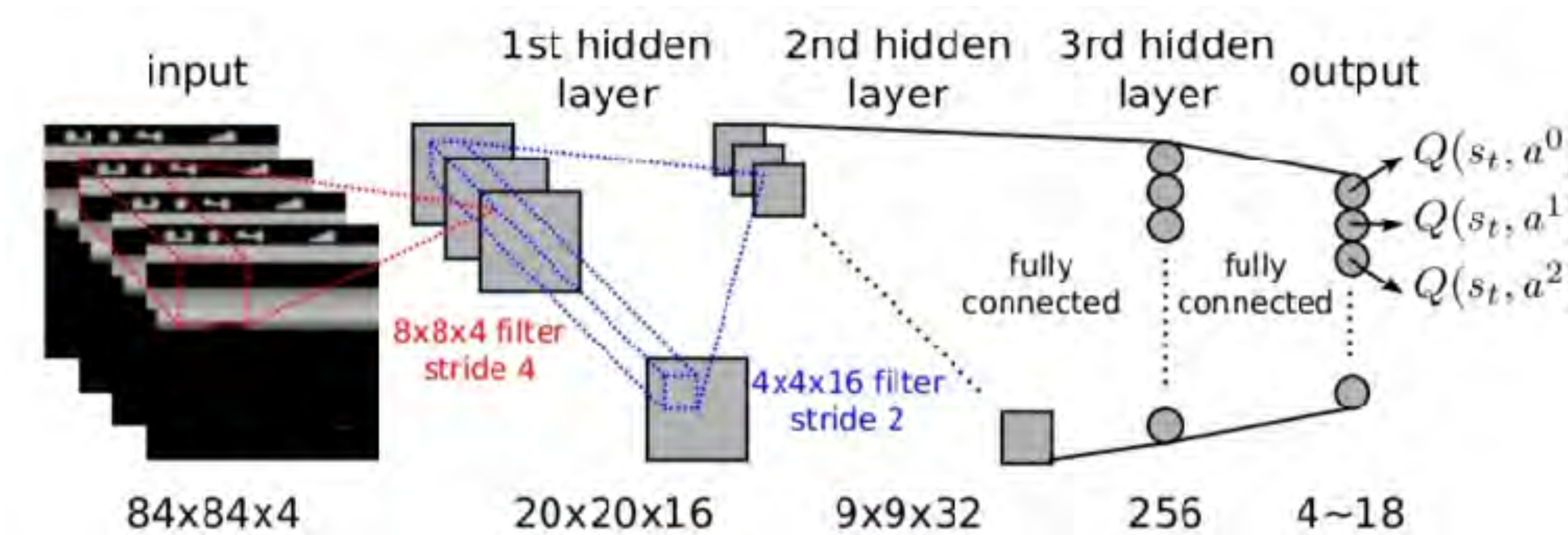
$$\begin{array}{ll} \text{minimize} & \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} x_t^* Q x_t + u_t^* R u_t\right] \\ \text{s.t.} & x_{t+1} = Ax_t + Bu_t + e_t \end{array}$$

"Greedy strategy": Build control $u_t = Kx_t$

- **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:** $K \leftarrow K - \alpha_t J(\tau)\, \nu$

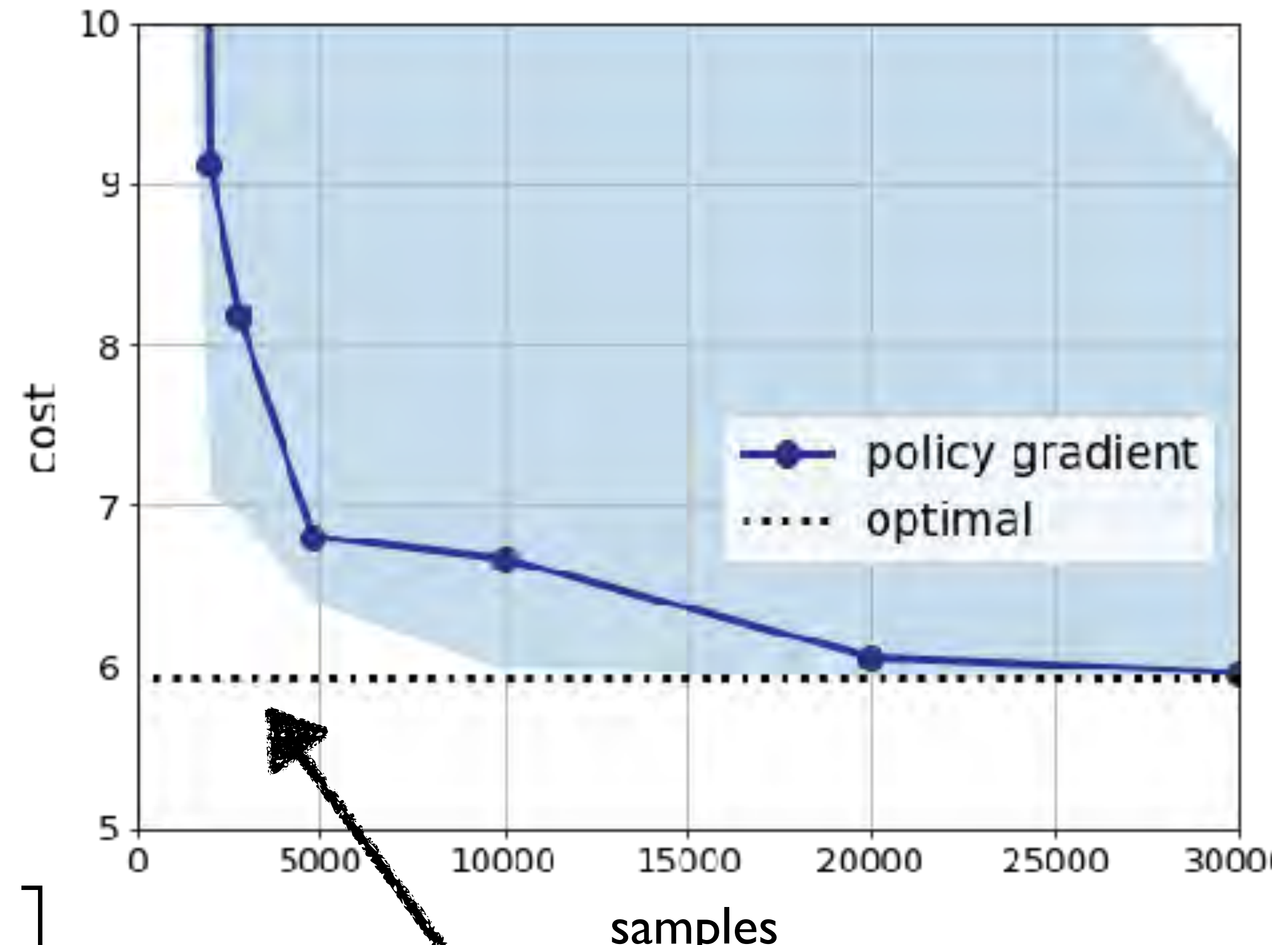
- Reinforce is NOT Magic
  - What is the variance?
  - Necessarily becomes derivative free as you are accessing the decision variable by sampling
  - Approximation can be far off
- But it’s certainly super easy!

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



nominal control and ADP
with 10 samples

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

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

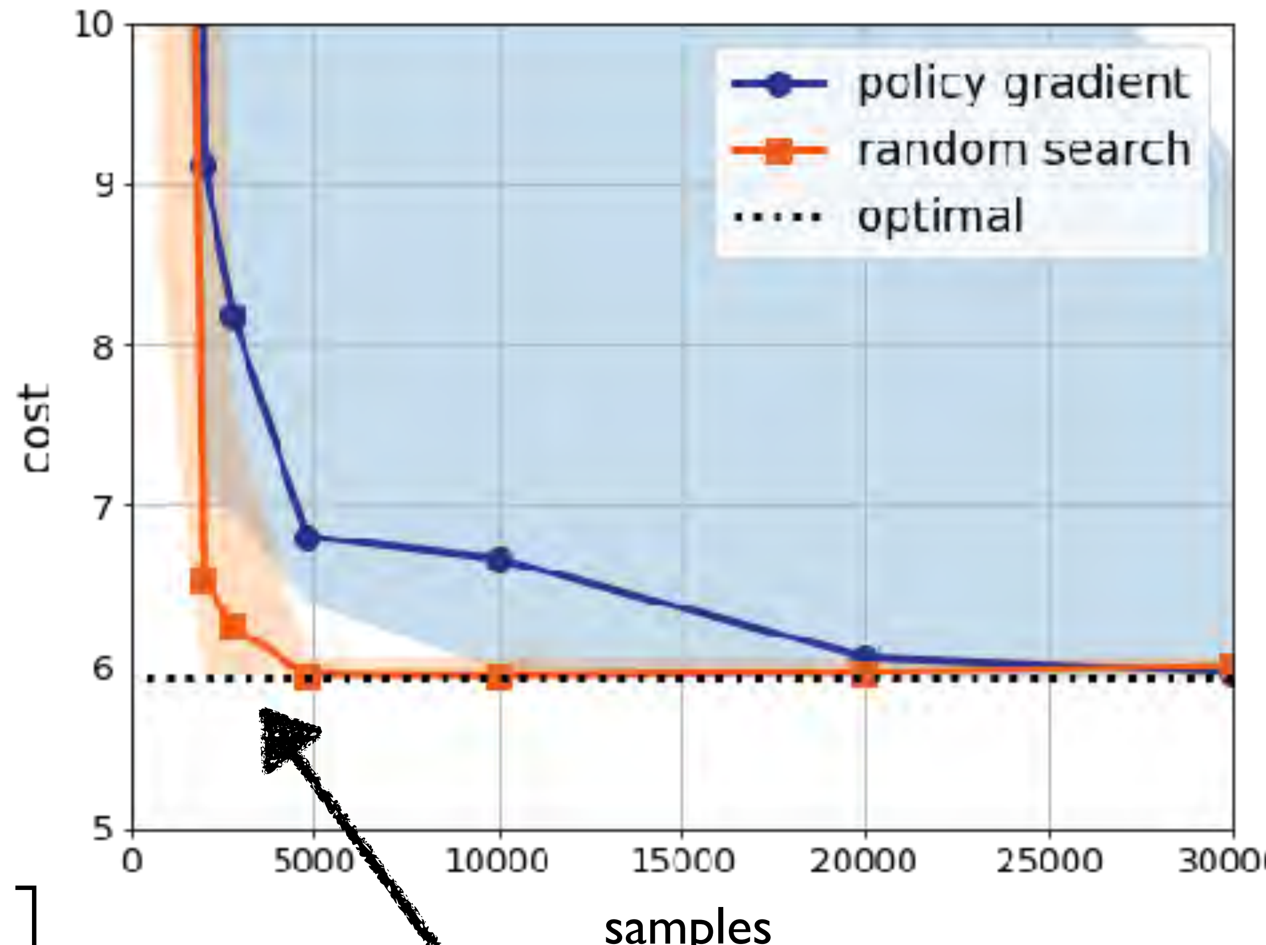
$$x_t = \begin{bmatrix} z_t \\ v_t \end{bmatrix}$$

“Simplest” Example: LQR

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

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

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



nominal control and ADP
with 10 samples

Extraordinary Claims Require Extraordinary Evidence*

* only if your prior is correct

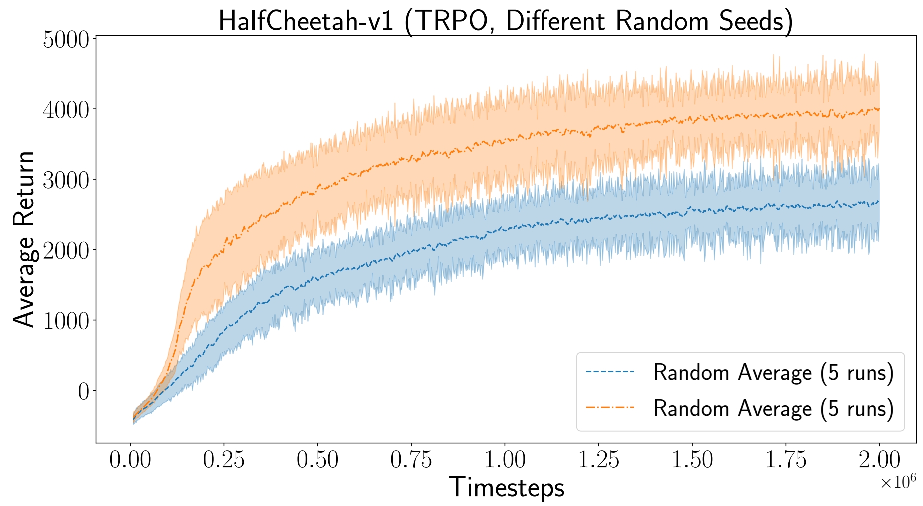
"How can we dismiss an entire field which claims such success?"

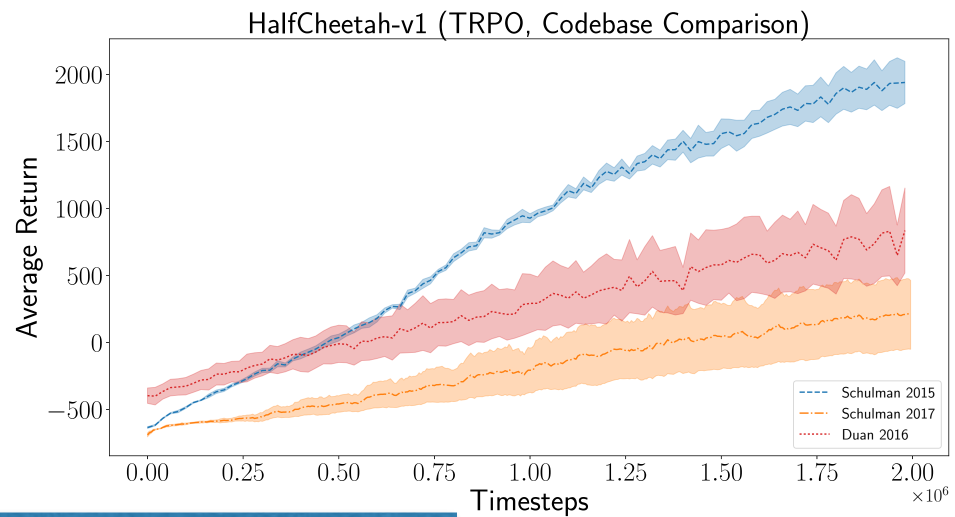
`blog.openai.com/openai-baselines-dqn/`

"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."

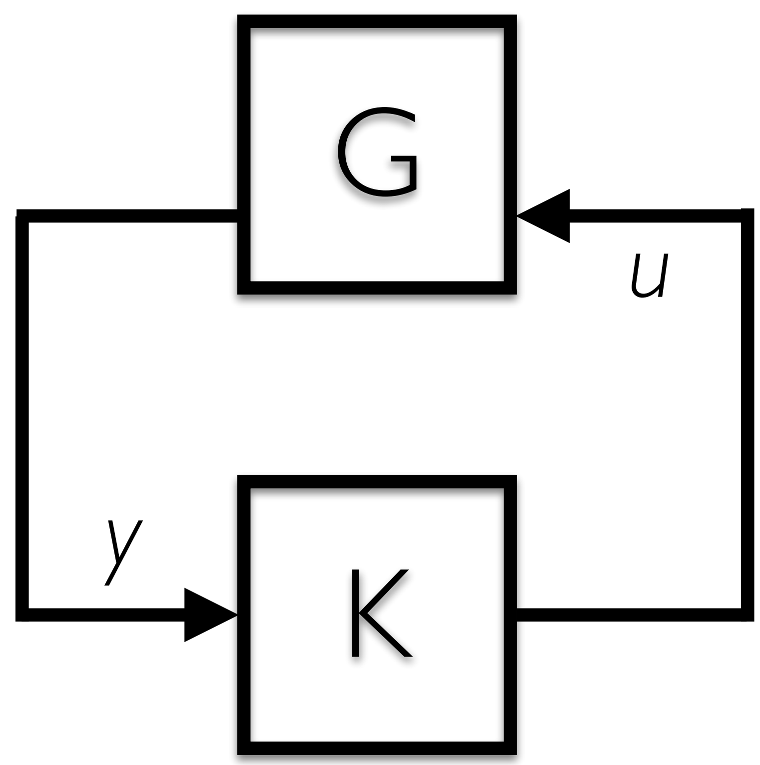
`arxiv:1709.06560`



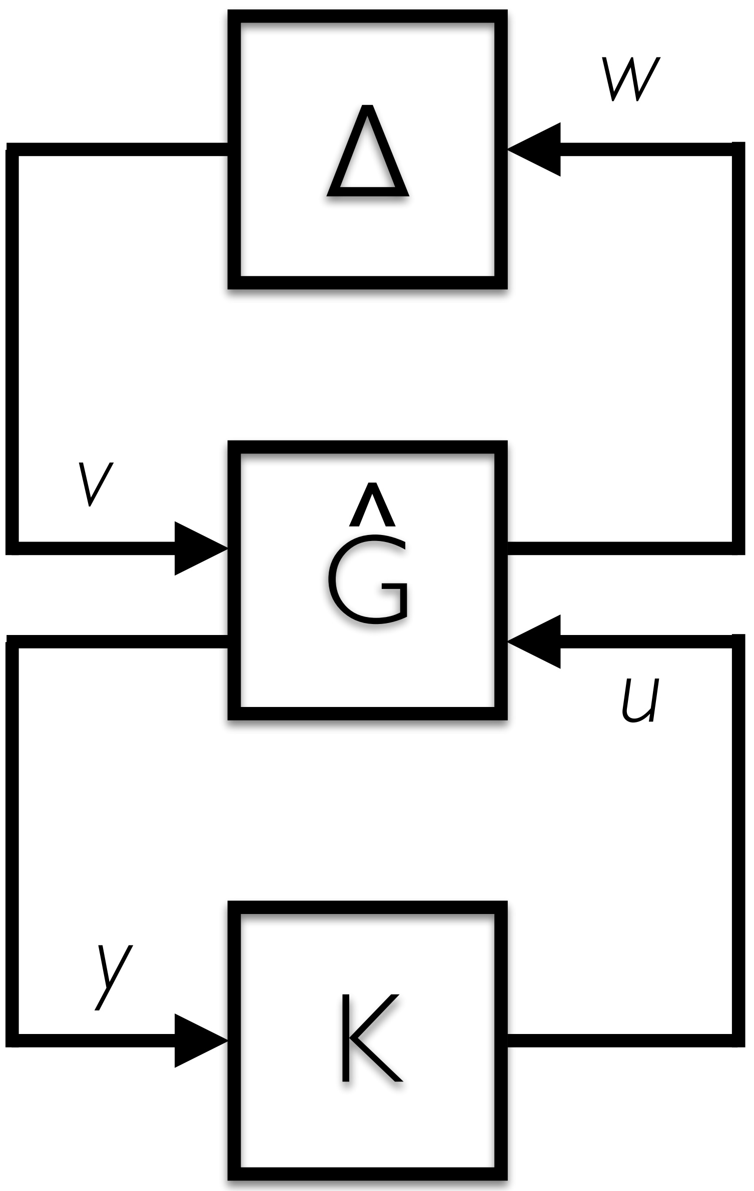


There has to be a better way!

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)

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

$\boxed{B \textit{ unknown!}}$

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

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

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

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

$\Downarrow$

Robust optimization problem:

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

Coarse-ID control (static case)

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

$\boxed{B \textit{ unknown!}}$

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

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

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: $\{(x_i, u_i)\}$ $\quad x_i = Bu_i + x_0 + e_i$

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

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)

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

B unknown!

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

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

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

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

Coarse-ID optimization

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & g(x;\vartheta) \le 0 \end{array}$$

ϑ *unknown!*

Collect data: $\{(x_i, g(x_i;\vartheta))\}$

Estimate ϑ: $\Rightarrow$ $\hat{\vartheta}$

Guarantee: $\mathrm{dist}(\vartheta, \hat{\vartheta}) \le \epsilon$ with high probability

Relaxation:
$$\begin{array}{ll} \underset{x}{\text{minimize}} & \hat{f}_{\text{robust}}(x) \\ \text{subject to} & g(x;\hat{\vartheta}) \le 0 \end{array}$$

Generalization bound $\mathrm{f}(\hat{x}) \le \mathrm{f}(x_\star) + \mathrm{err}\left(\epsilon, \|f - \hat{f}_{\text{robust}}\|, x_\star, \vartheta\right)$

"Simple" Example: LQR

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

Gaussian noise

"Obvious strategy": Estimate (A,B), build control $u_t = Kx_t$

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]

"Simple" Example: LQR

"Obvious strategy": Estimate $(\hat{A}, \hat{B})$, build control $u_t = \hat{K} x_t$

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

Solving an SDP relaxation of this robust control problem yields

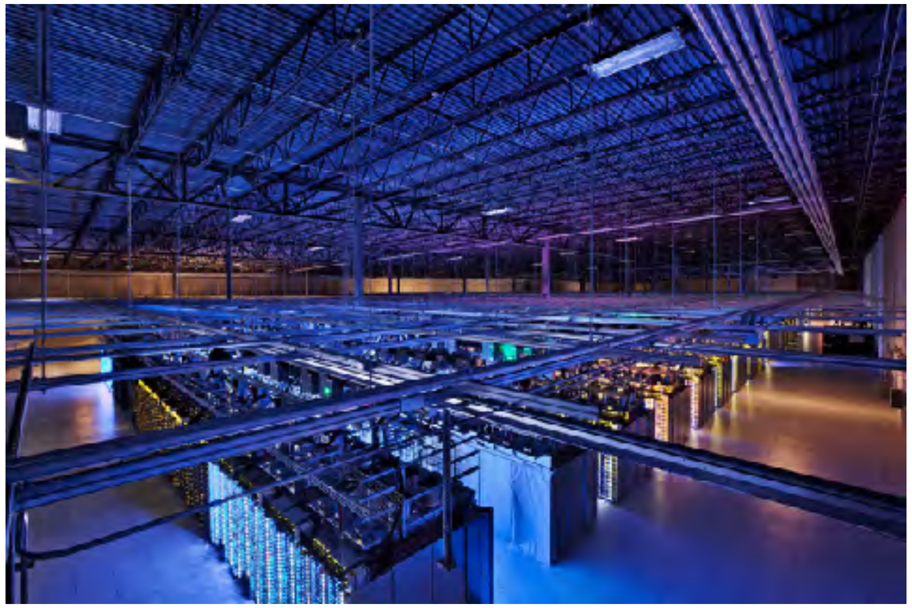
$$\frac{J(\hat{K}) - J_\star}{J_\star} \le C\, \Gamma_{\mathrm{cl}} \left(\lambda_{\min}(\Lambda_c)^{-1/2} + \|K_\star\|_2 \right) \sqrt{\frac{\sigma^2 (d+p)}{T}} \quad \text{w.h.p.}$$

$\Lambda_c = A \Lambda_c A^* + BB^*$
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!

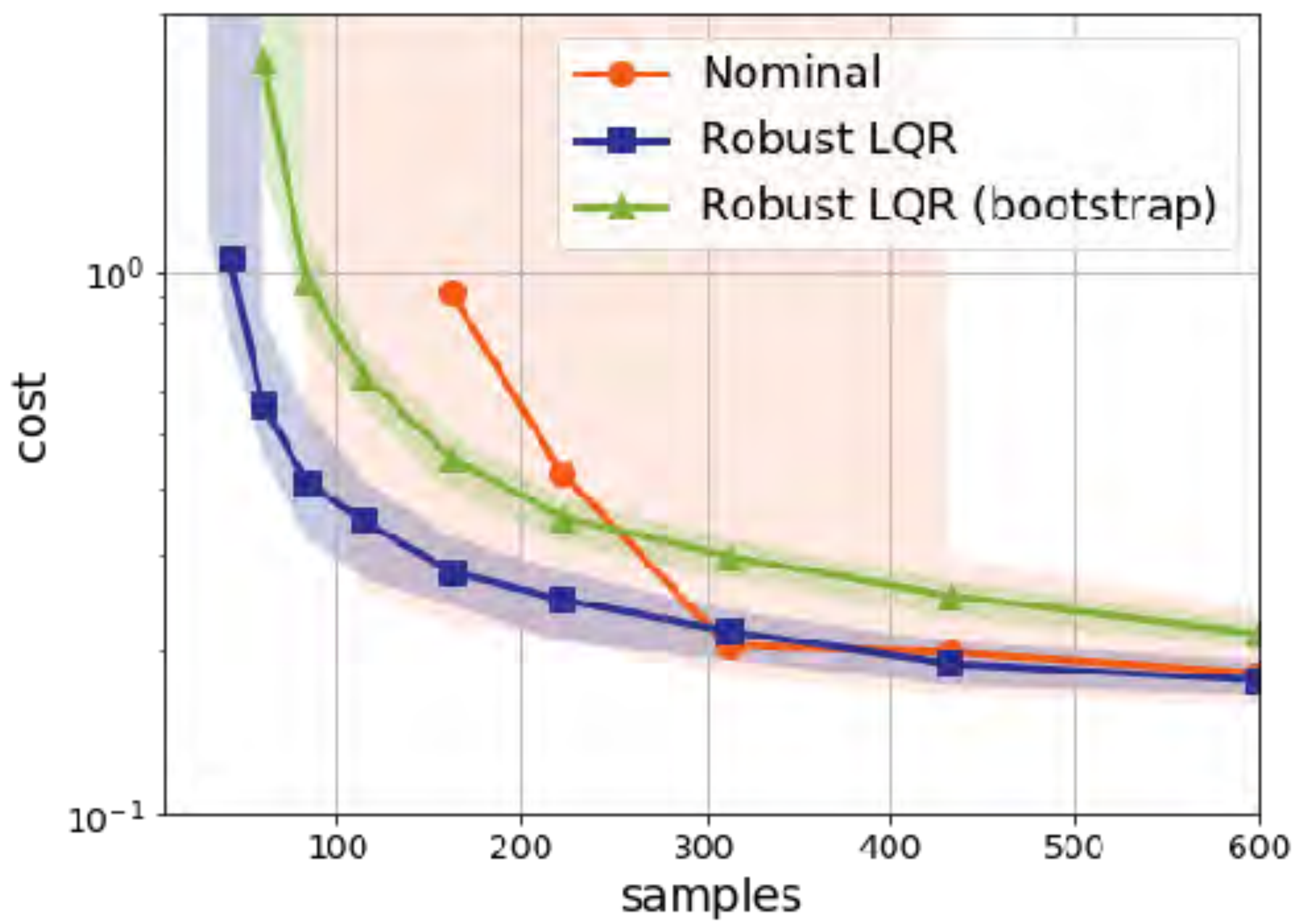
[Dean, Mania, Matni, R., Tu 2017]

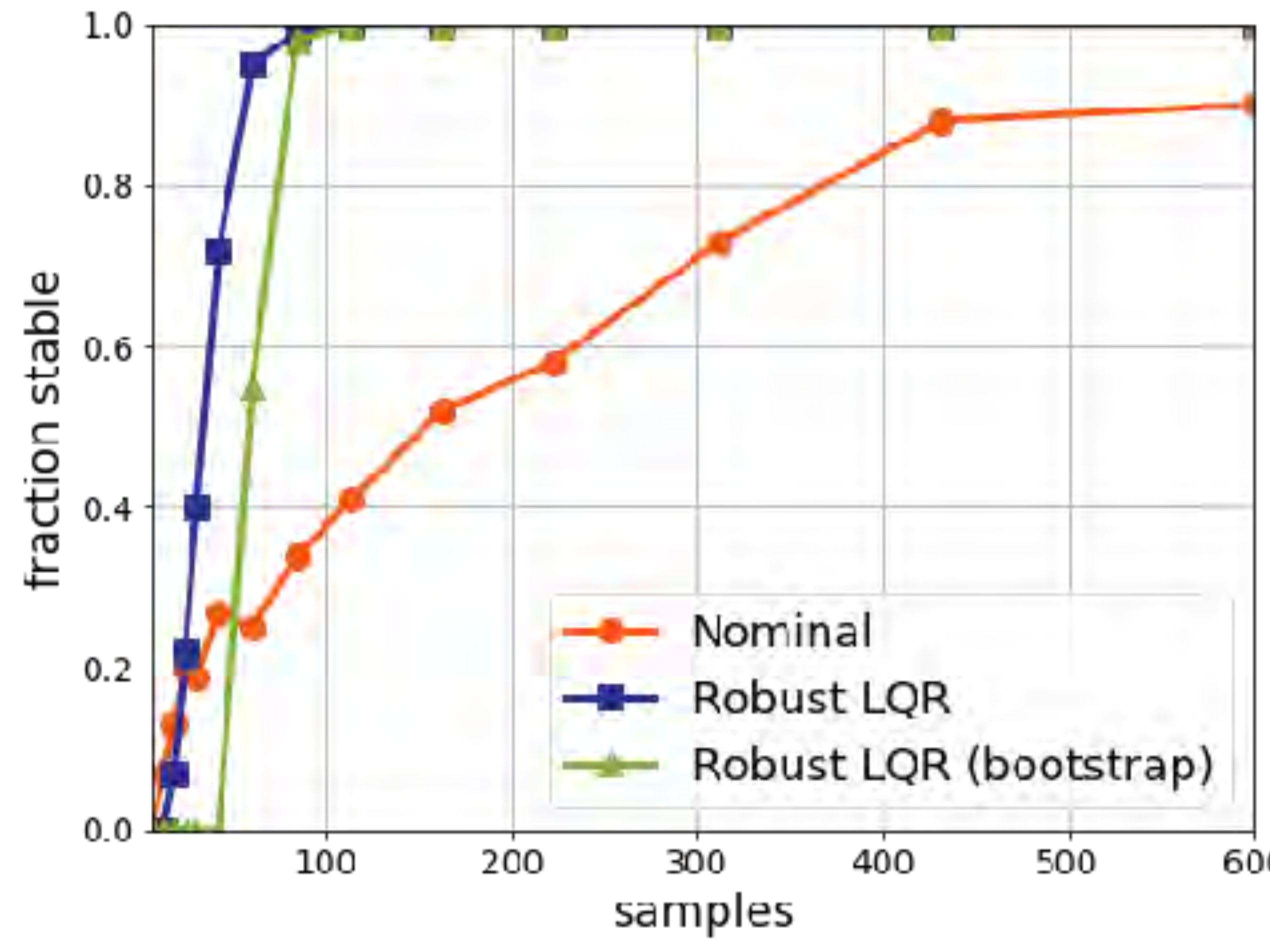


Why robust?

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

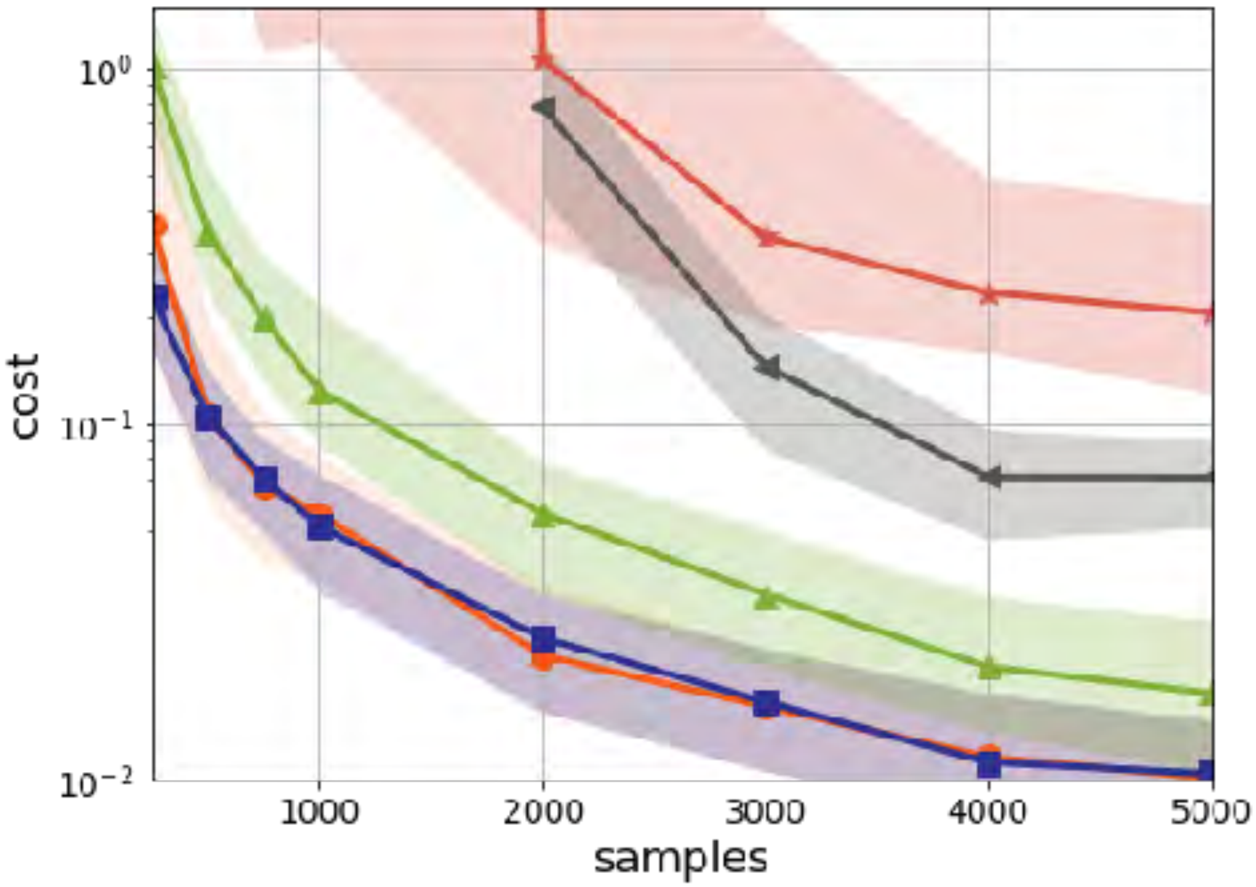
Slightly unstable system, system ID tends to think some nodes are stable

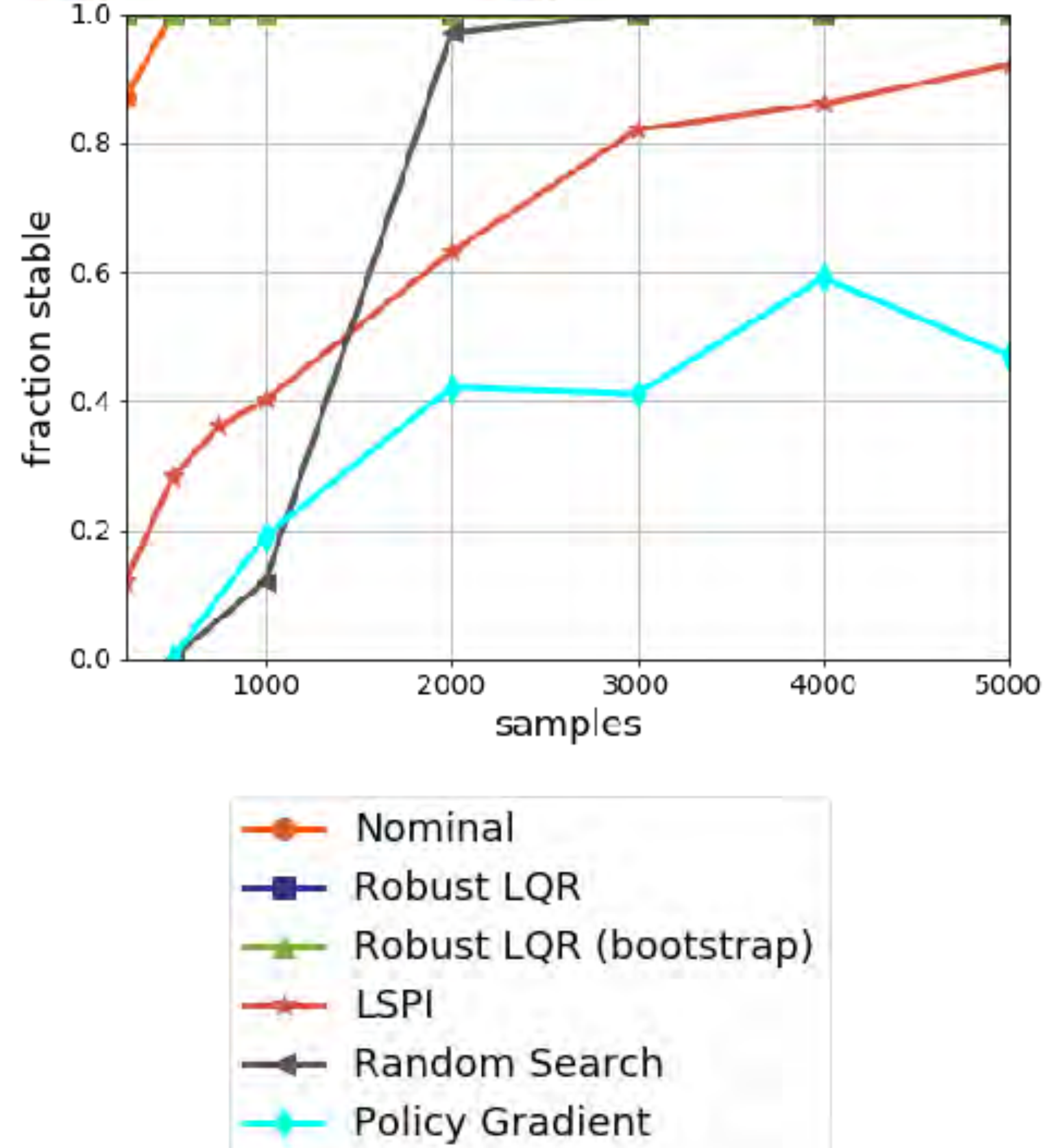




Least-squares estimate may yield unstable controller

Robust synthesis yields stable controller





Model-free performs worse than model-based

Why has no one done this before?

- Our guarantees for least-squares estimation required some heavy machinery.
  - Indeed, best bounds building on papers from last few years
- Our SDP relaxation uses brand new techniques in controller parameterization (Matni et al)
  - Naive robust synthesis is nonconvex and requires solving very large SDPs
- *The Singularity has arrived!*

Even LQR is not simple!!!

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

s.t. $\quad x_{t+1} = A x_t + B u_t + e_t$ *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* $\qquad \Gamma_{\mathrm{cl}} := \|(zI - A - BK_\star)^{-1}\|_{\mathcal{H}_\infty}$ *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.

“Simplest” Example: LQR

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

How many samples are needed for near optimal control?

- Lots of asymptotic work in the 80s (adaptive control)
- **Fietcher, 1997:** PAC, discounted costs, many assumptions on contractivity, bugs in proof.
- **Abbas-Yadkori & Szepesvári, 2011:** Regret, exponential in dimension, no guarantee on parameter convergence, NP-hard subroutine.
- **Ibrahimi *et al*, 2012:** require sparseness in state transitions
- **Ouyang *et al*, 2017:** Bayesian setting, unrealistic assumptions, not implementable
- **Abeille and Lazaric , 2015:** Suboptimal bounds

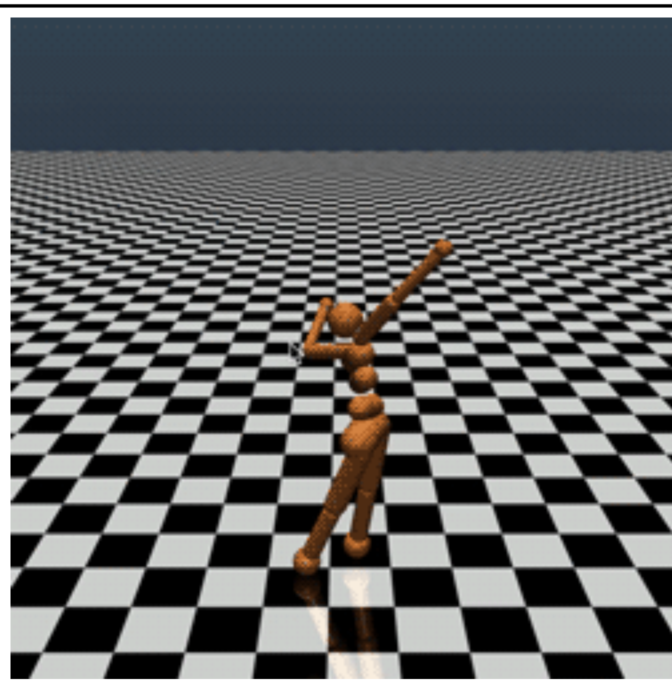
The Linearization Principle

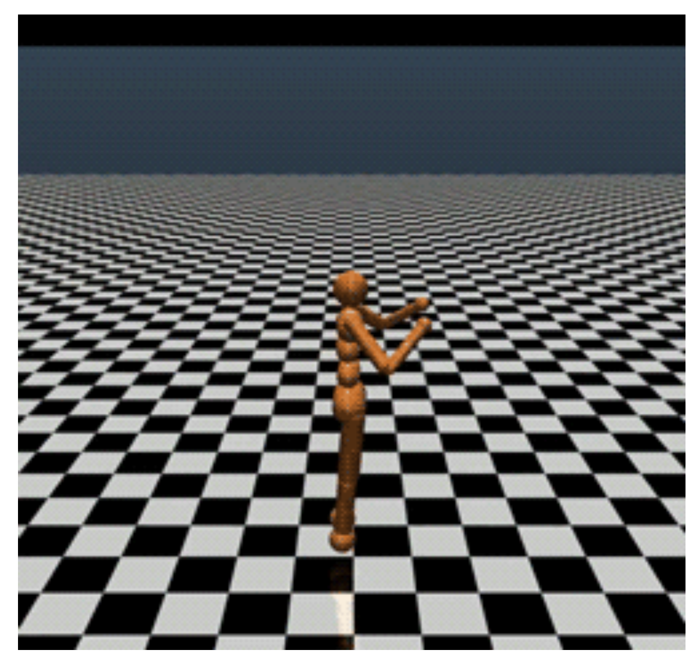
If a machine learning algorithm does crazy things when restricted to linear models, it's going to do crazy things on complex nonlinear models too.

What happens when we return to nonlinear models?

Random search of linear policies outperforms Deep Reinforcement Learning

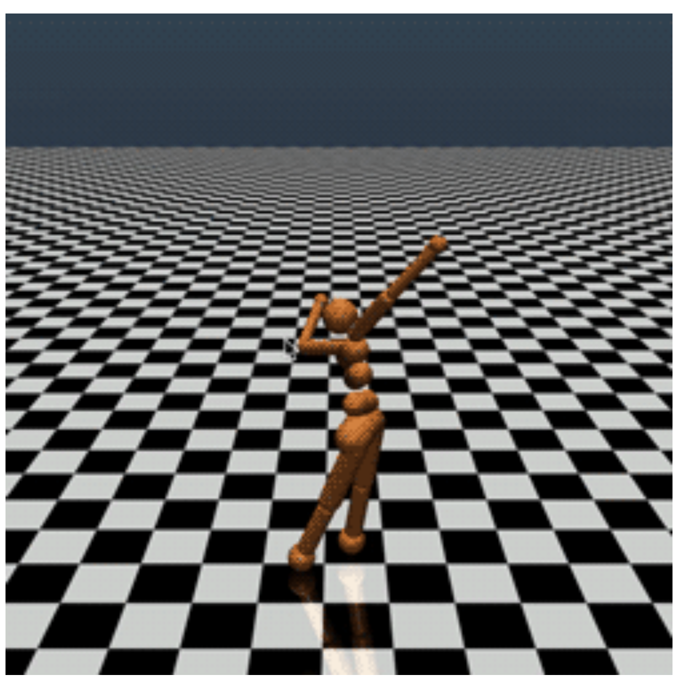
Larger is better		**Maximum average reward**		
Task	**RS**	**NG-lin**	**NG-rbf**	**TRPO-nn**
Swimmer-v1	365	366	365	131
Hopper-v1	3909	3651	3810	3668
HalfCheetah-v1	6722	4149	6620	4800
Walker	11389	5234	5867	5594
Ant	5146	4607	4816	5007
Humanoid	11600	6440	6849	6482

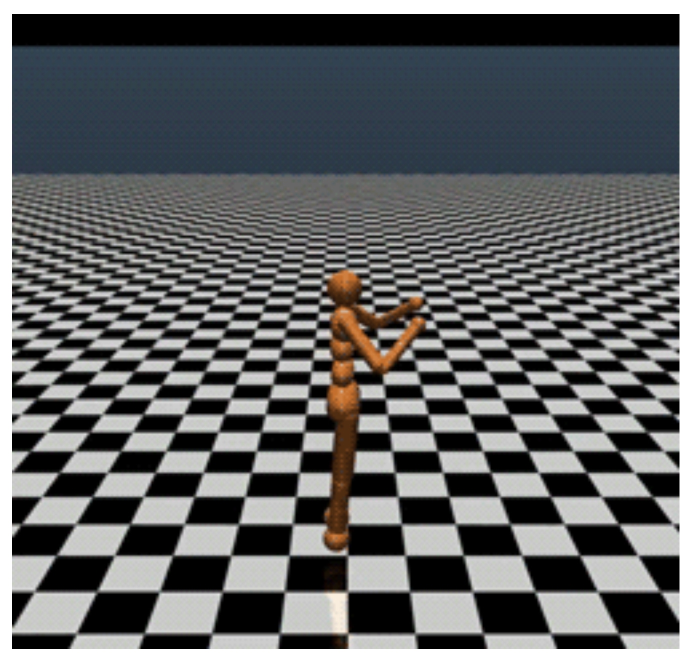


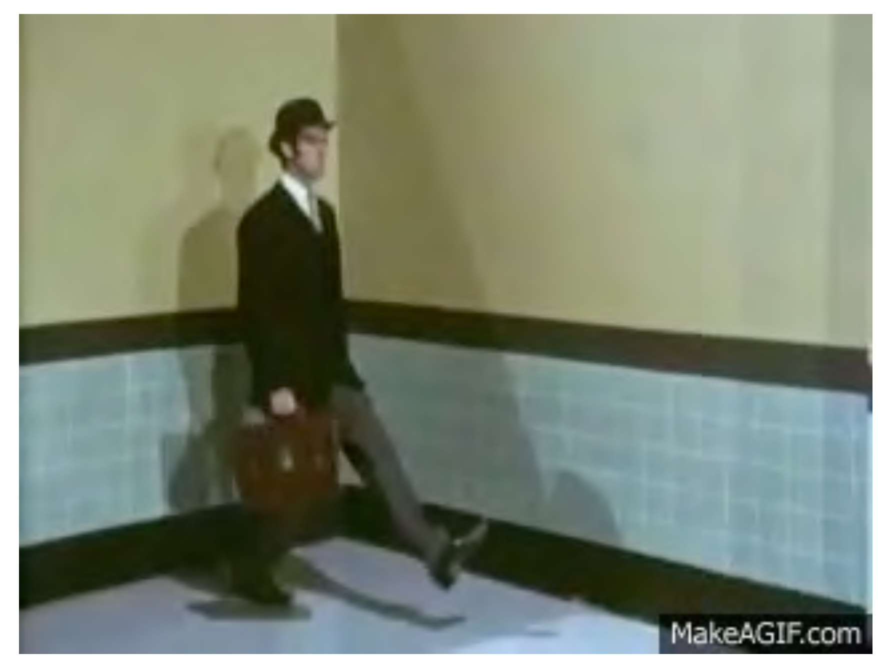


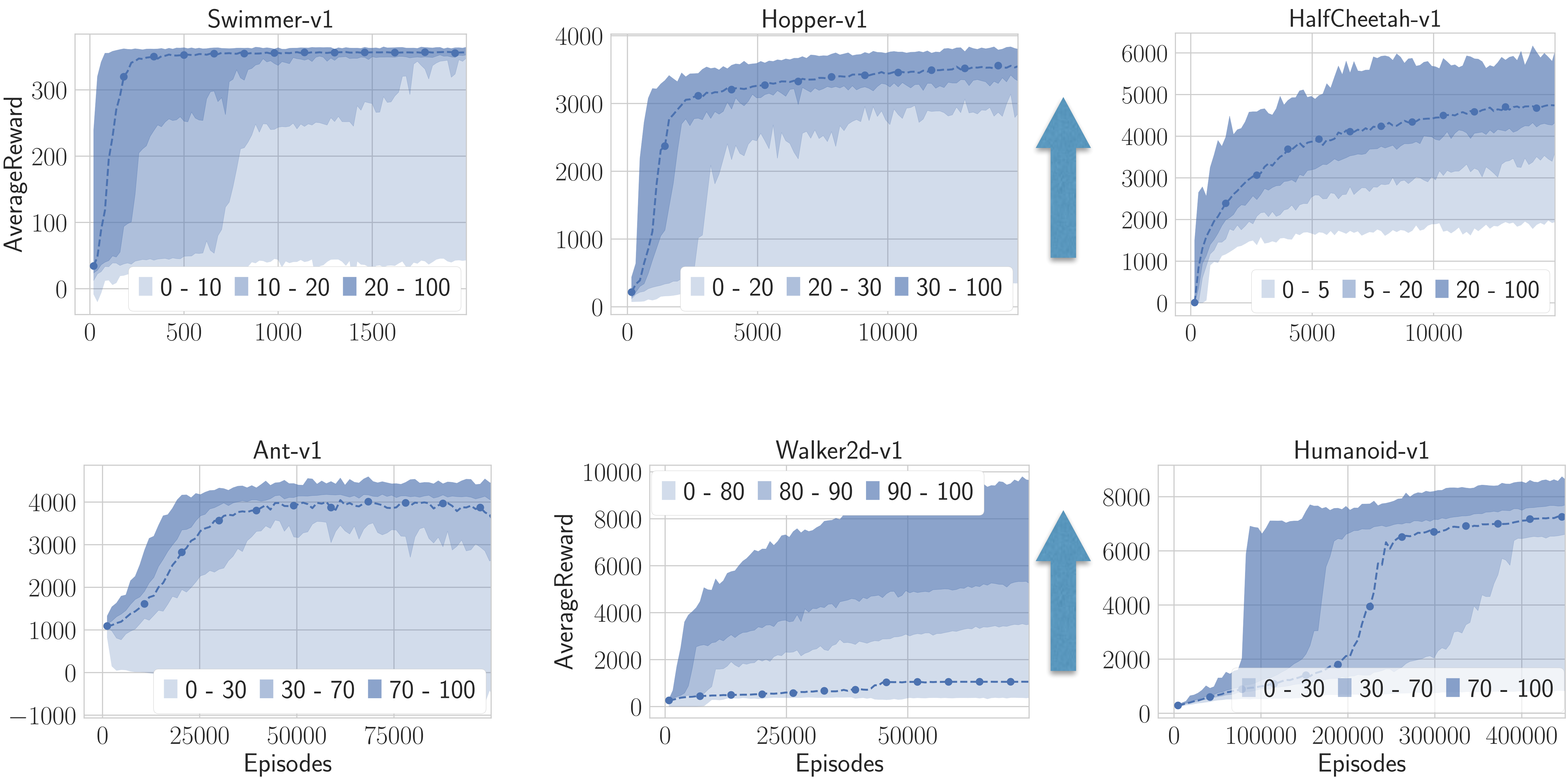
Larger is better

		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

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

Optimal Policy:

$$\boxed{\pi_k(\tau_k) = \arg\min_u C_k(x_k, u) + \mathbb{E}_e\left[V_{k+1}(f_k(x_k, u, e))\right]}$$

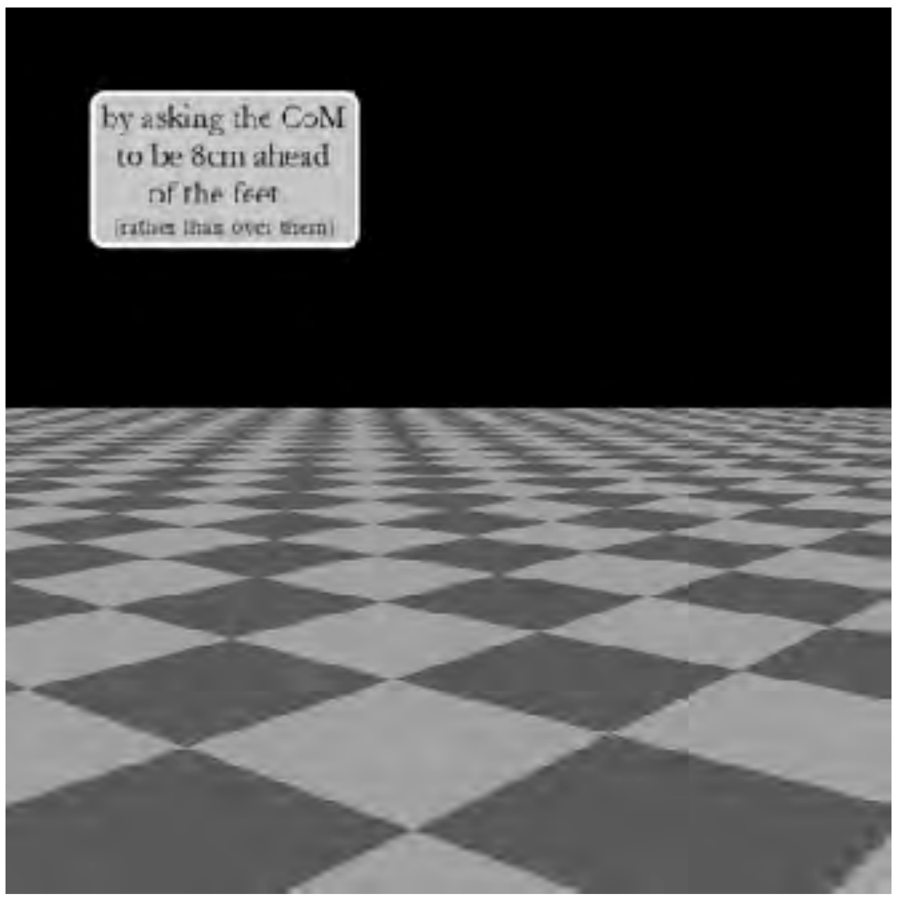
MPC: use the policy at every time step

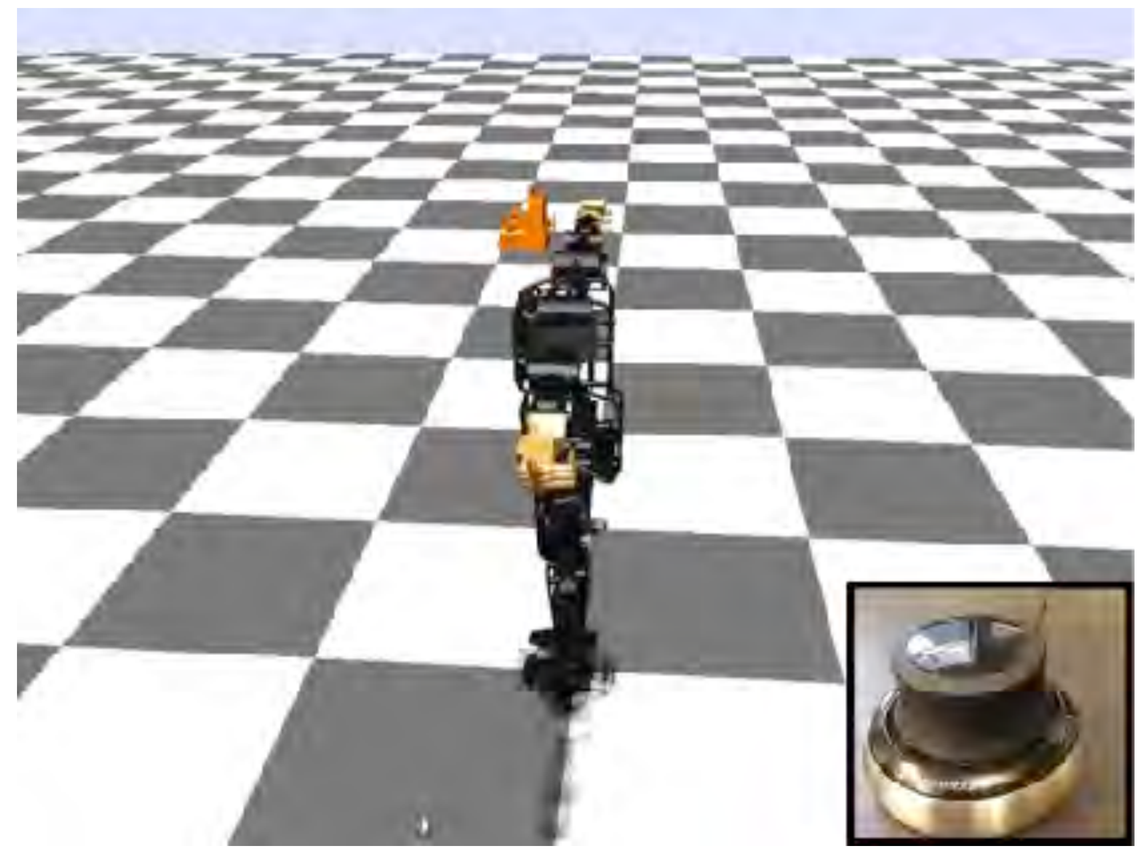
$$\boxed{\pi_1(x) = \arg\min_u C_k(x, u) + \mathbb{E}_e\left[V_1(f_1(x, u, e))\right]}$$

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

MPC trades improved computation for control cleverness, requiring significant planning for each action.

Model Predictive Control





Videos from Todorov Lab

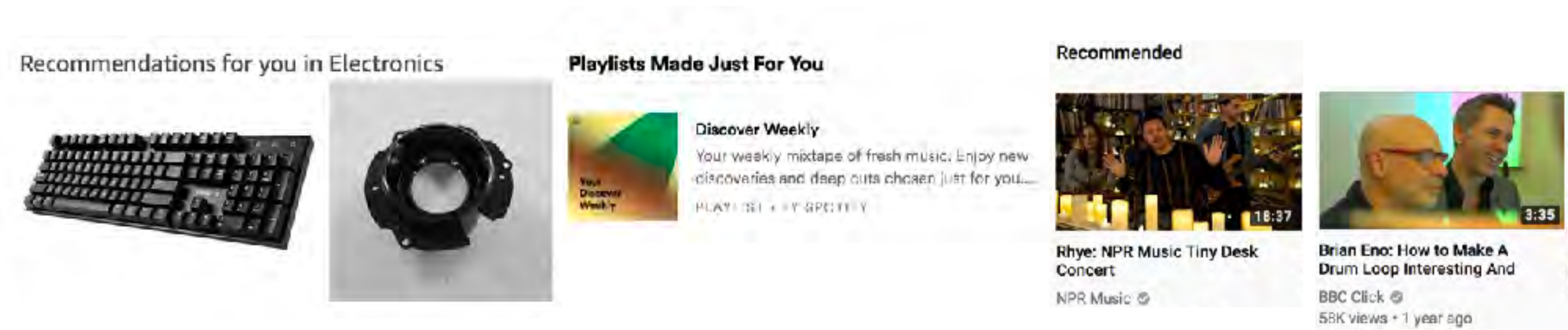
`https://homes.cs.washington.edu/~todorov/`

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

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.

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`
- B. Recht "A short and biased tour of reinforcement learning." Coming soon…

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