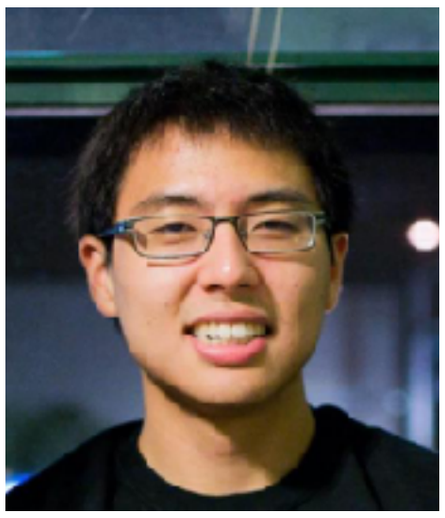
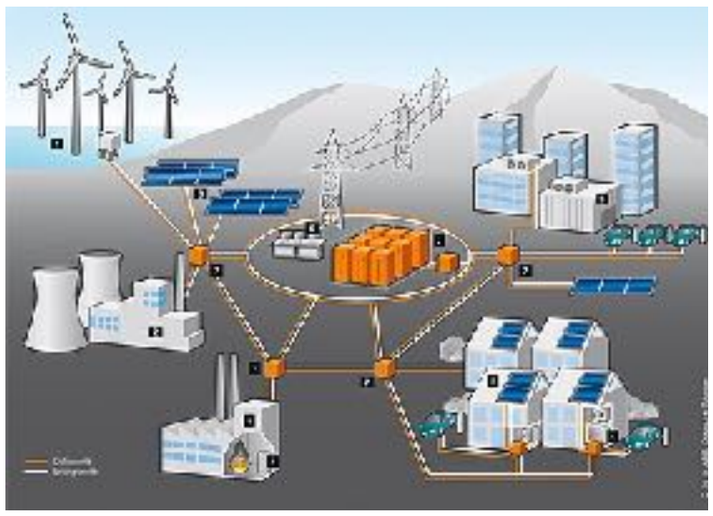
Collaborators





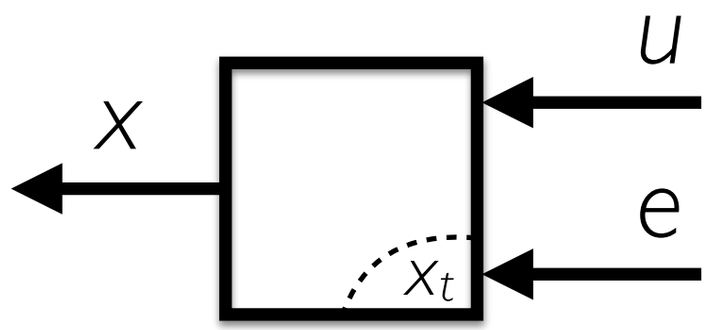
Joint work with Sarah Dean, Aurelia Guy, Horia Mania, Nikolai Matni, Max Simchowitz, and Stephen Tu.



trustable, scalable, predictable

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.*

x_t is the state, u_t is the input, e_t is a noise process

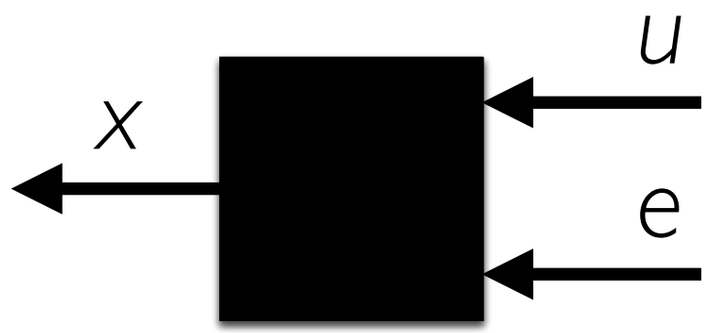
f_t is the state-transition function

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

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

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*.

x_t is the state, u_t is the input, 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.

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

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

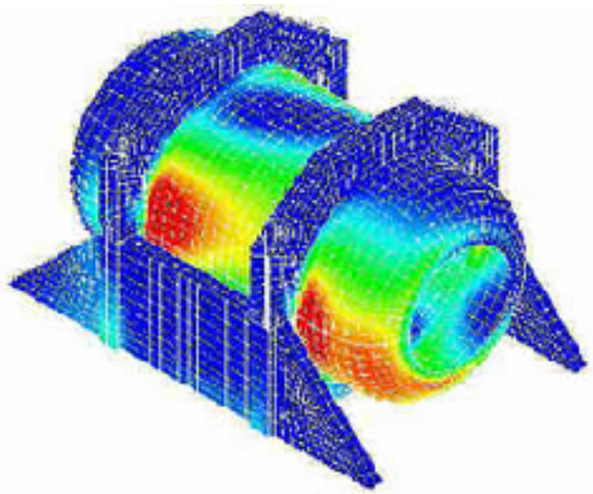
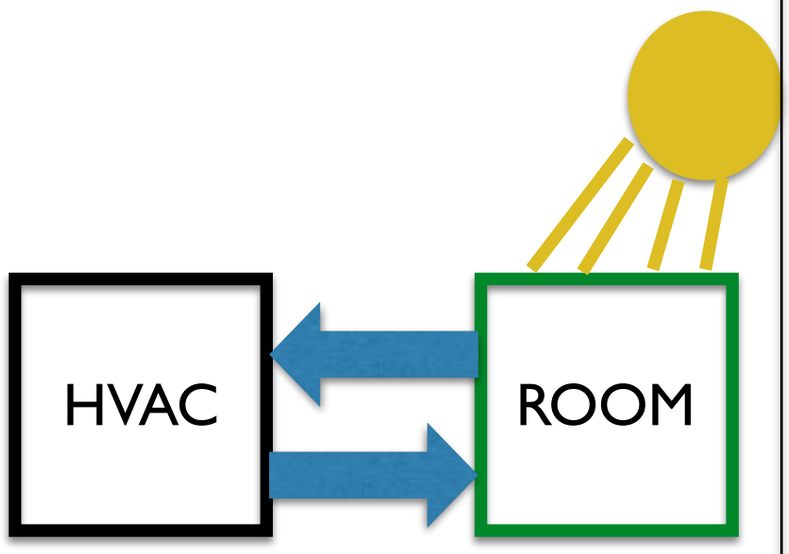
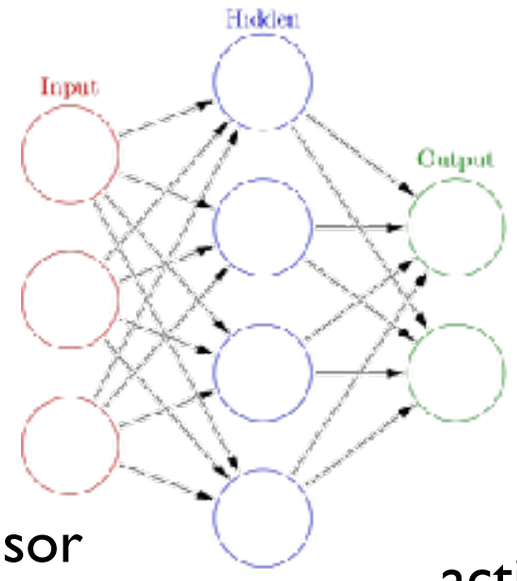


Figure 1 http://www.noranenz.com

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



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



sensor
state

action

Identify everything

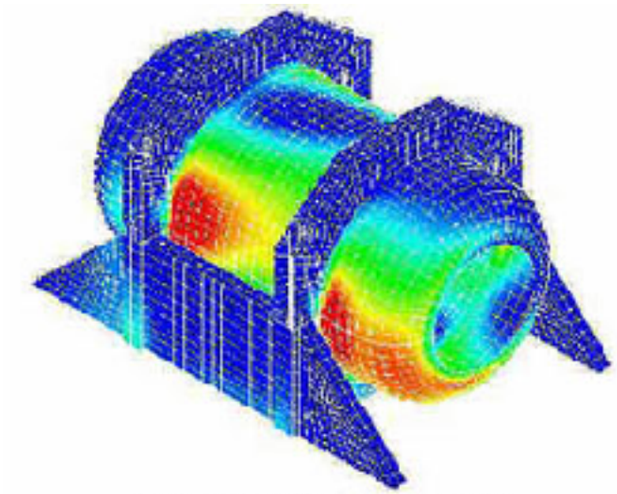
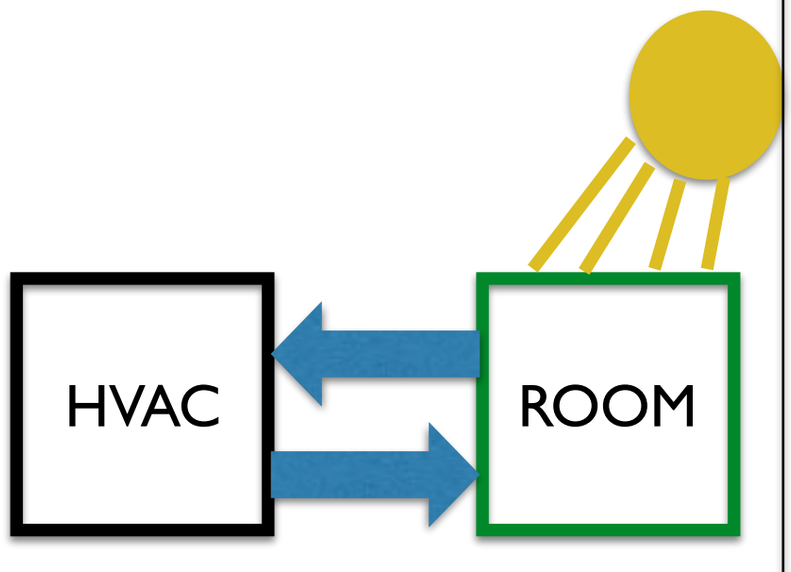


Figure 1 http://www.noranengr.com

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

- High performance aerodynamics

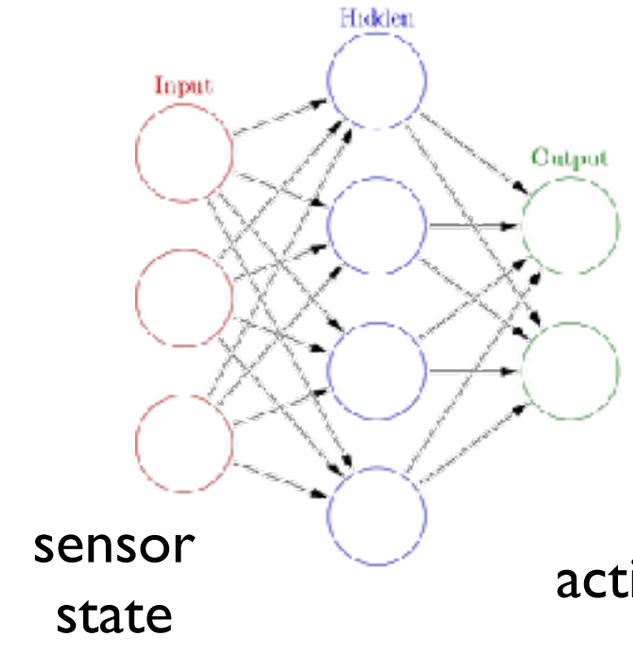
Identify a coarse model



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

- model predictive control

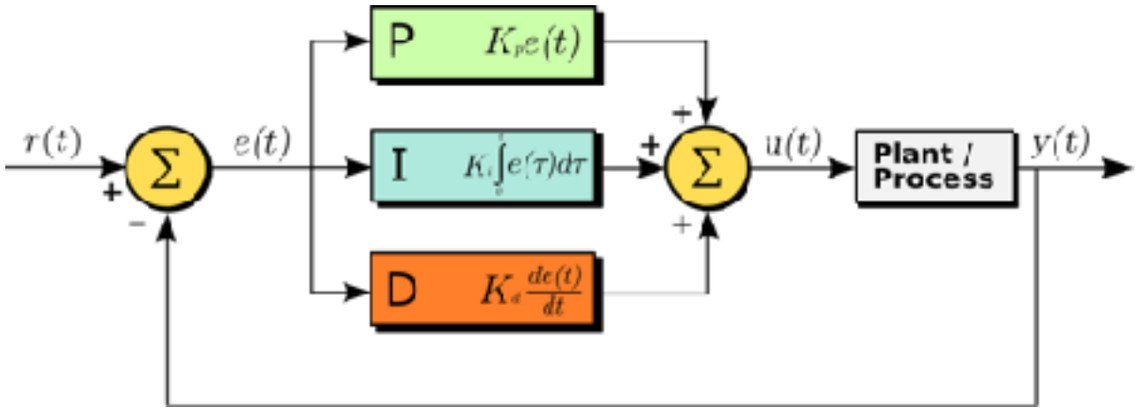
We don't need no stinking models!

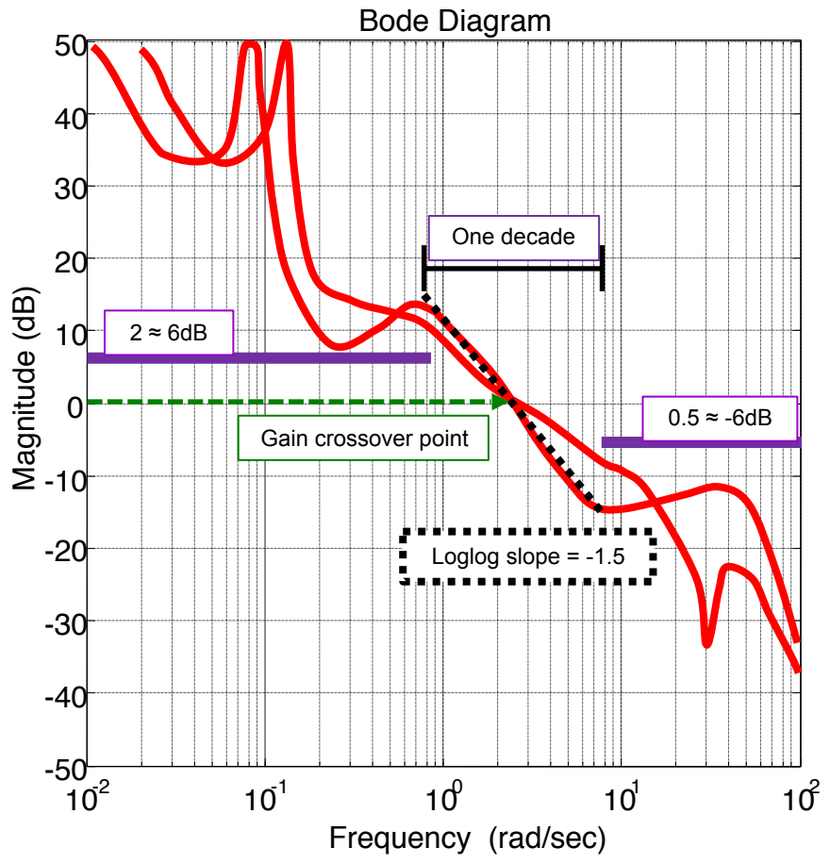


- reinforcement learning
- PID control?

We need robust fundamentals to distinguish these approaches

But PID control works…





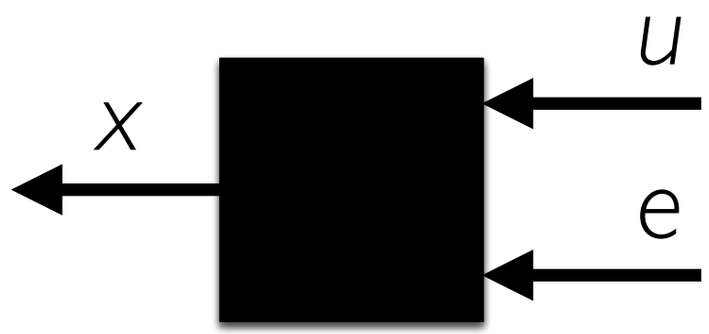
2 parameters suffice for 95% of all control applications.

How much needs to be modeled for more advanced control?

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

Learning to control

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



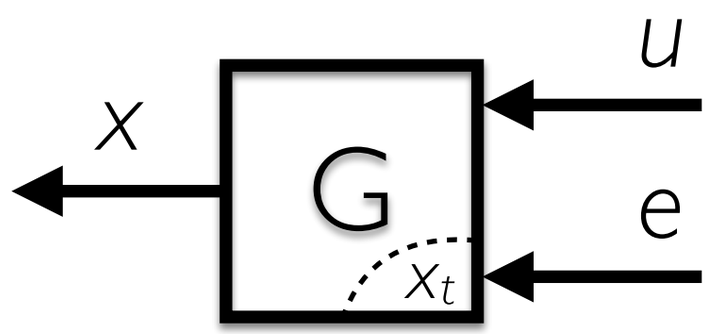
Oracle: You can generate N trajectories of length T.

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?

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**
  - **Direct policy search:** search for actions from data
- **Approximate dynamic programming**
  - estimate cost from data (with or without model)

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

Model-based RL

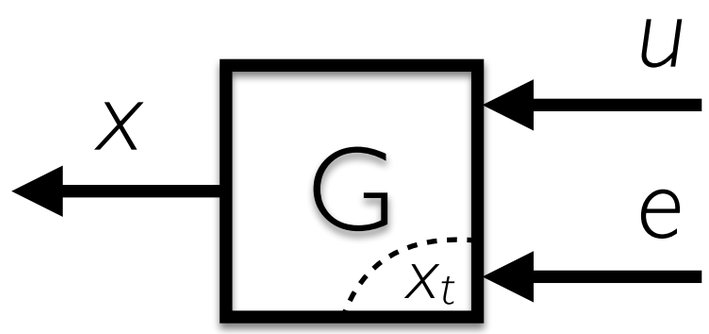
Collect some simulation data. Should have $x_{t+1} \approx \varphi(x_t, u_t) + \nu_t$

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

Solve approximate problem:

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

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**
  - **Direct policy search:** search for actions from data
- **Approximate dynamic programming**
  - estimate cost from data (with or without model)

"Simplest" Example: LQR

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

- Optimization simplicity
- Elegant Dynamic Programming solutions
- Exact solution for baseline
- Static state feedback solution on infinite horizon
- Natural robustness
- Broadly applicable as is
- Core of many MPC and nonlinear control methods

"Simplest" Example: LQR

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

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?

Sample Complexity?

Continuous Control:

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

ADP

model-based

policy search

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

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

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

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

s.t. $x_{t+1} = Ax_t + Bu_t + e_t$ $\quad x \in \mathbb{R}^d$

$u_t = Kx_t$ $\quad u \in \mathbb{R}^p$

ADP vs model-based

Fix state feedback K, How to estimate value function? $V(x) = x^* P_\star x$

Estimate closed-loop model, solve DLE: $\hat{P}_{\mathrm{plug}}$

Use ADP (LSTD): $\hat{P}_{\mathrm{lstd}}$

Thm: $\lim_{T\to\infty} T\,\mathbb{E}\left[\left\|\hat{P}_{\mathrm{plug}} - P_\star\right\|_F^2\right] \le \lim_{T\to\infty} T\,\mathbb{E}\left[\left\|\hat{P}_{\mathrm{lstd}} - P_\star\right\|_F^2\right]$

There exist instances where

$\lim_{T\to\infty} T\,\mathbb{E}\left[\left\|\hat{P}_{\mathrm{plug}} - P_\star\right\|_F^2\right] \le O(d^2)$ $\qquad$ $\lim_{T\to\infty} T\,\mathbb{E}\left[\left\|\hat{P}_{\mathrm{lstd}} - P_\star\right\|_F^2\right] \ge \Omega(d^3)$

and no algorithm can do better than $O(d^2)$.

[Tu, R. 2018]

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

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

$u_t = Kx_t$

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

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

Direct Policy Search vs model-based

Find state feedback K. How does its cost compare to optimal $J_\star$?

Estimate state-space model, solve DARE.

Run *policy gradient*.

There exist instances where $d>p$ and

$$\lim_{N\to\infty} N\,\mathbb{E}\left[J(\hat{K}_{\mathrm{plug}})/J_\star - 1\right] = O(p)$$

$$\liminf_{N\to\infty} N\,\mathbb{E}\left[J(\hat{K}_{\mathrm{pg}})/J_\star - 1\right] \geq \Omega(T^2(p^4/d + d^2 p))$$

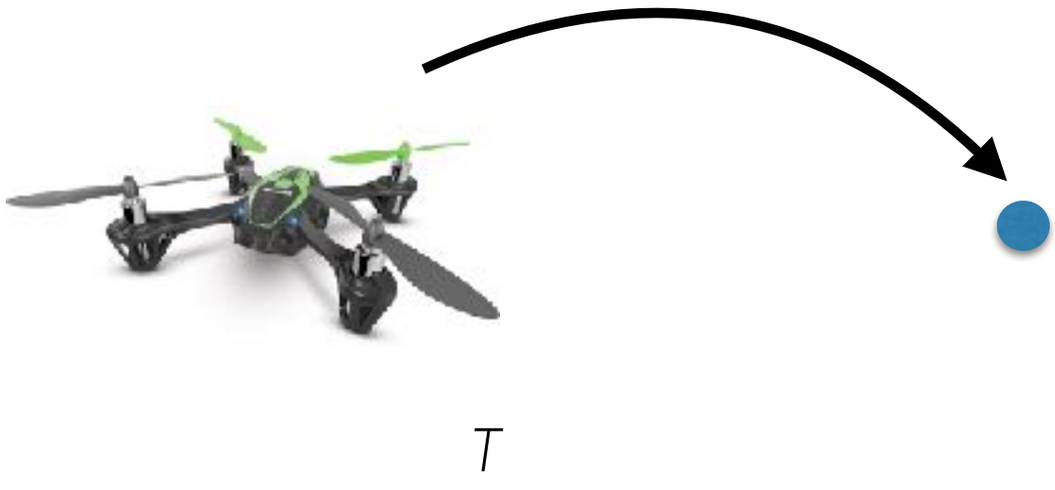
$$\liminf_{N\to\infty} N\,\mathbb{E}\left[J(\hat{K}_{\mathrm{pg}})/J_\star - 1\right] \geq \Omega(p^3/d + p^2)$$

(with a sophisticated uncomputable baseline)

and no algorithm can do better than $O(p^2/d)$.

[Tu, R. 2018]

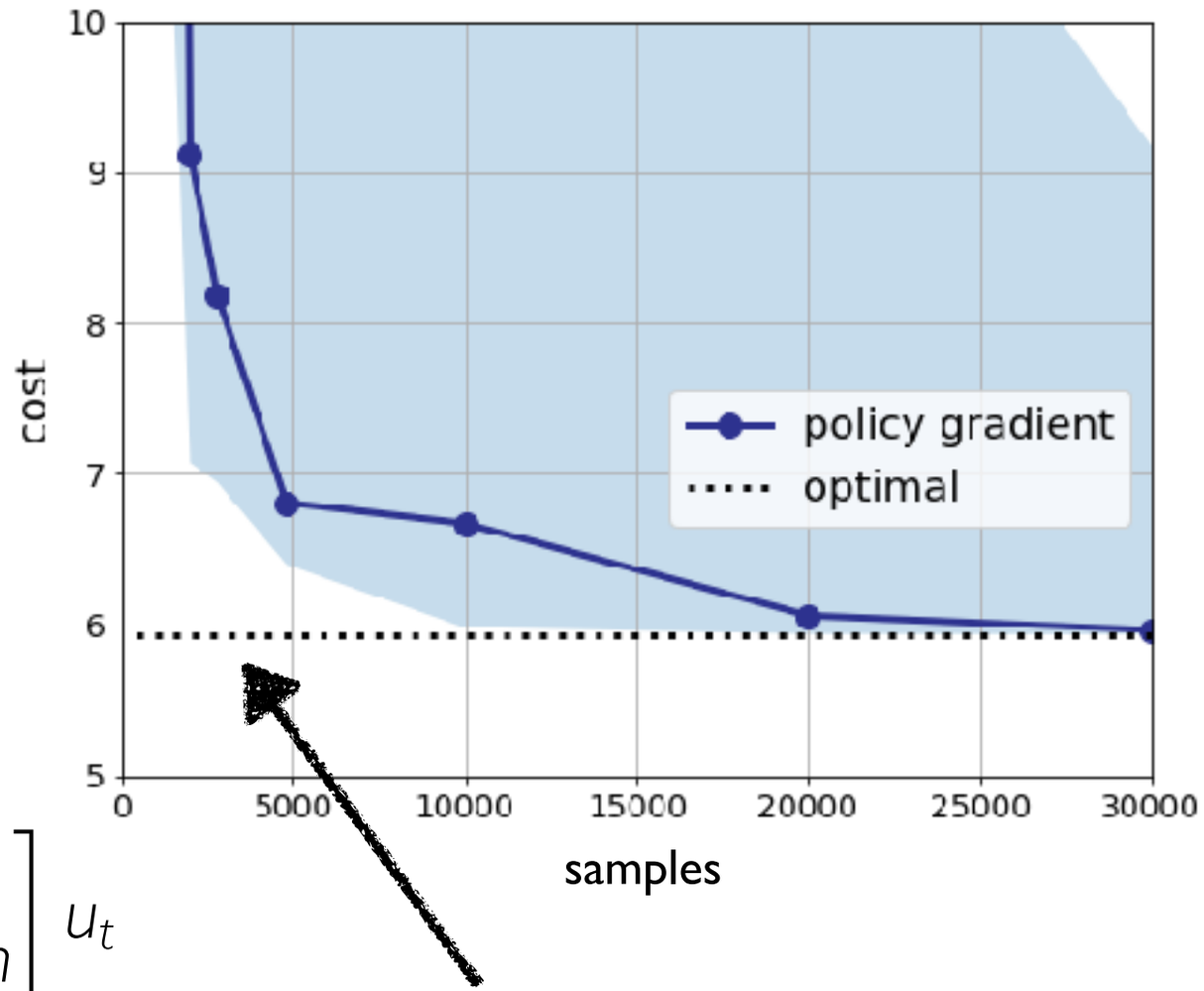
"Simplest" Example: LQR



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

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

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



nominal control with 10 samples

`argmin.net/code/lqr_policy_comparisons.ipynb`

Extraordinary Claims Require Extraordinary Evidence*

* only if your prior is correct

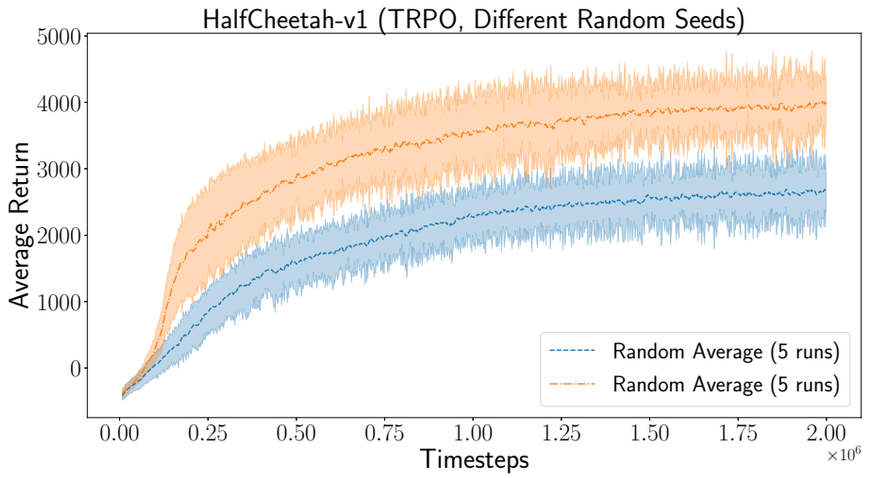
"How can we dismiss an entire field which claims such success?"

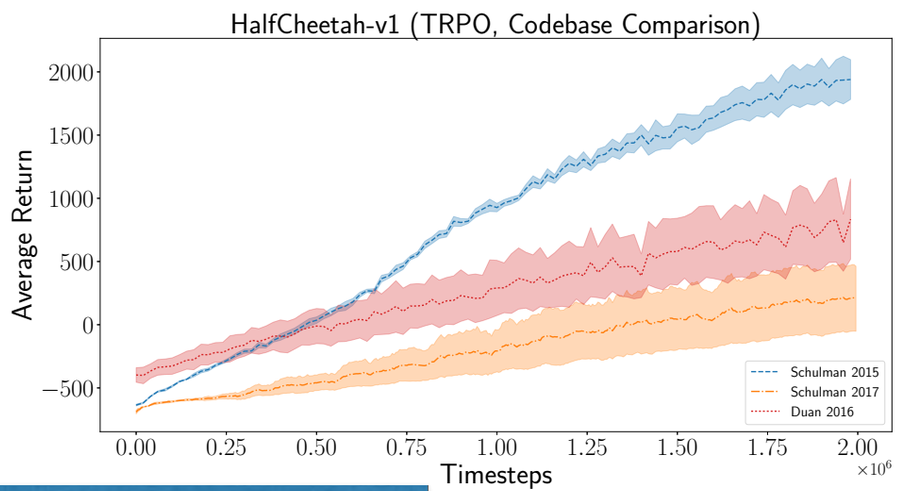
`blog.openai.com/openai-baselines-dqn/`

"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."

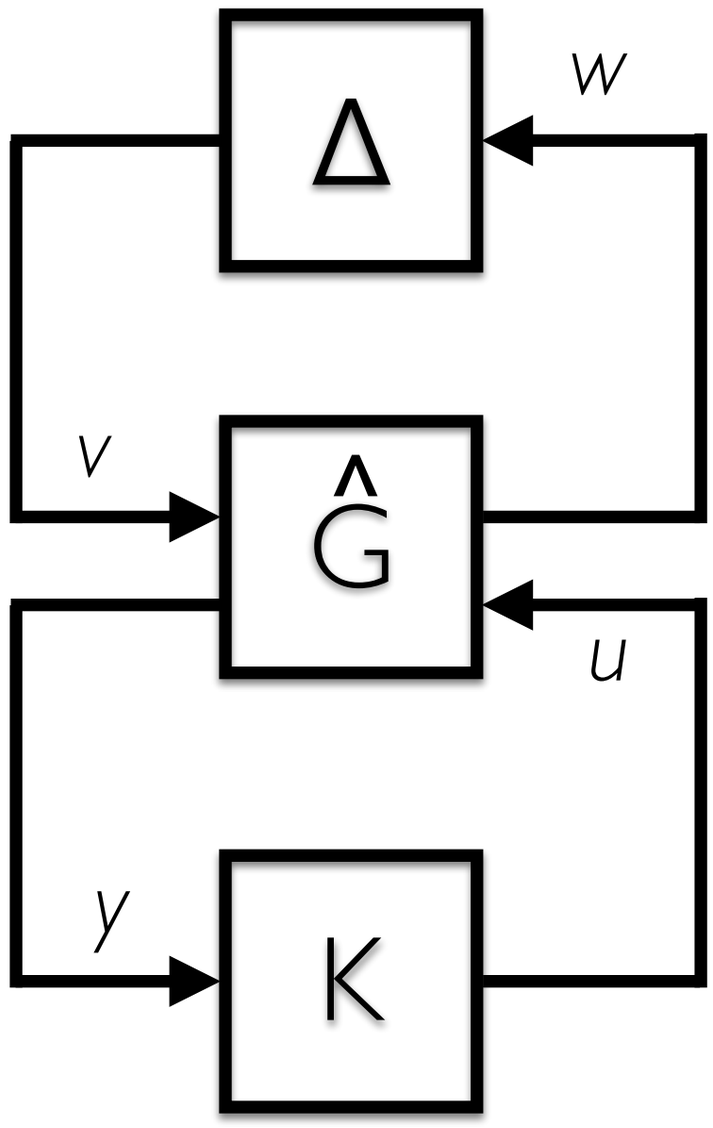
`arxiv:1709.06560`





There has to be a better way!

Coarse-ID control



High dimensional stats bounds the error

Coarse-grained model is trivial to fit

Design robust control for feedback loop

Robust certainty equivalence.

Coarse-ID Control for LQR

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

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

How many samples are needed to Estimate (A,B)? (A stable)

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 $T \geq \tilde{O}\left(\frac{\sigma^2(d+p)}{\epsilon^2}\right)$ then $\|A - \hat{A}\| \leq \epsilon$ and $\|B - \hat{B}\| \leq \epsilon$ w.h.p.

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

[Mania, Jordan, R., Simchowitz, Tu, 2018]

Coarse-ID Control for LQR

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

Solving an SDP relaxation of this robust control problem yields

$$\frac{J(\hat{K}) - J_\star}{J_\star} \le C \sqrt{\frac{\sigma^2 (d+p)}{T}} \qquad \text{w.h.p.}$$

$J(\hat{K})$ = LQR cost of coarse-ID control $\quad$ $J_\star$ = optimal LQR when (A,B) known

This also tells you when your cost is finite!

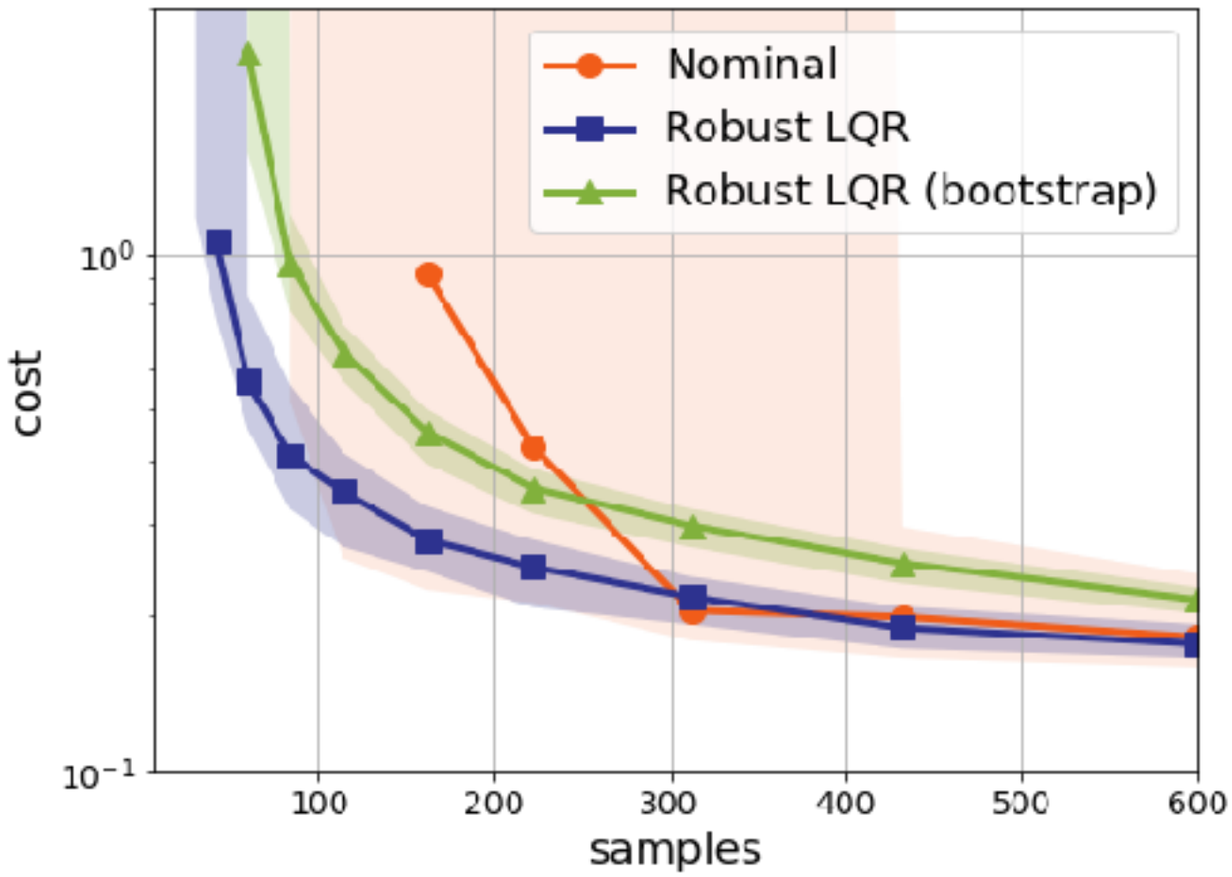
Robust LQR solution via robust system level synthesis

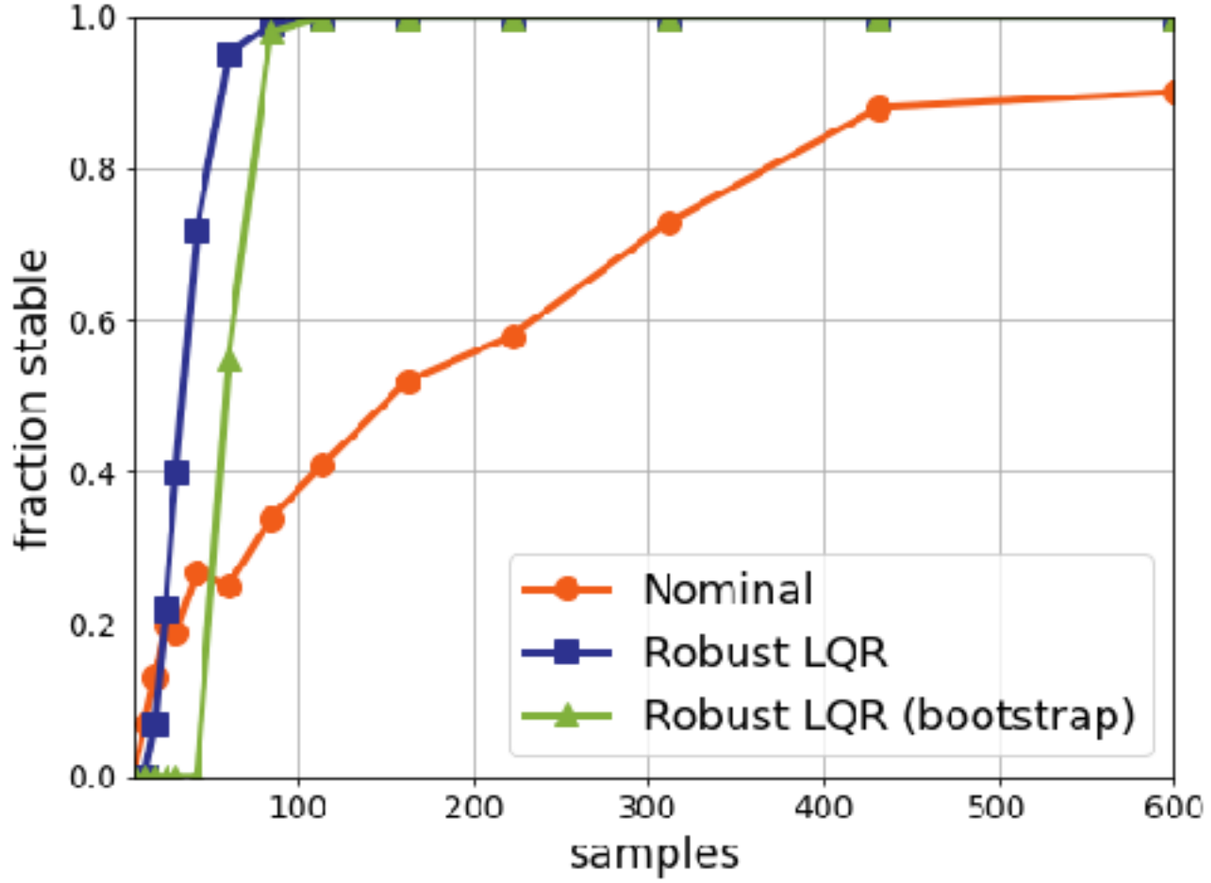
[Dean, Mania, Matni, R., Tu 2017]

Why robust?

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

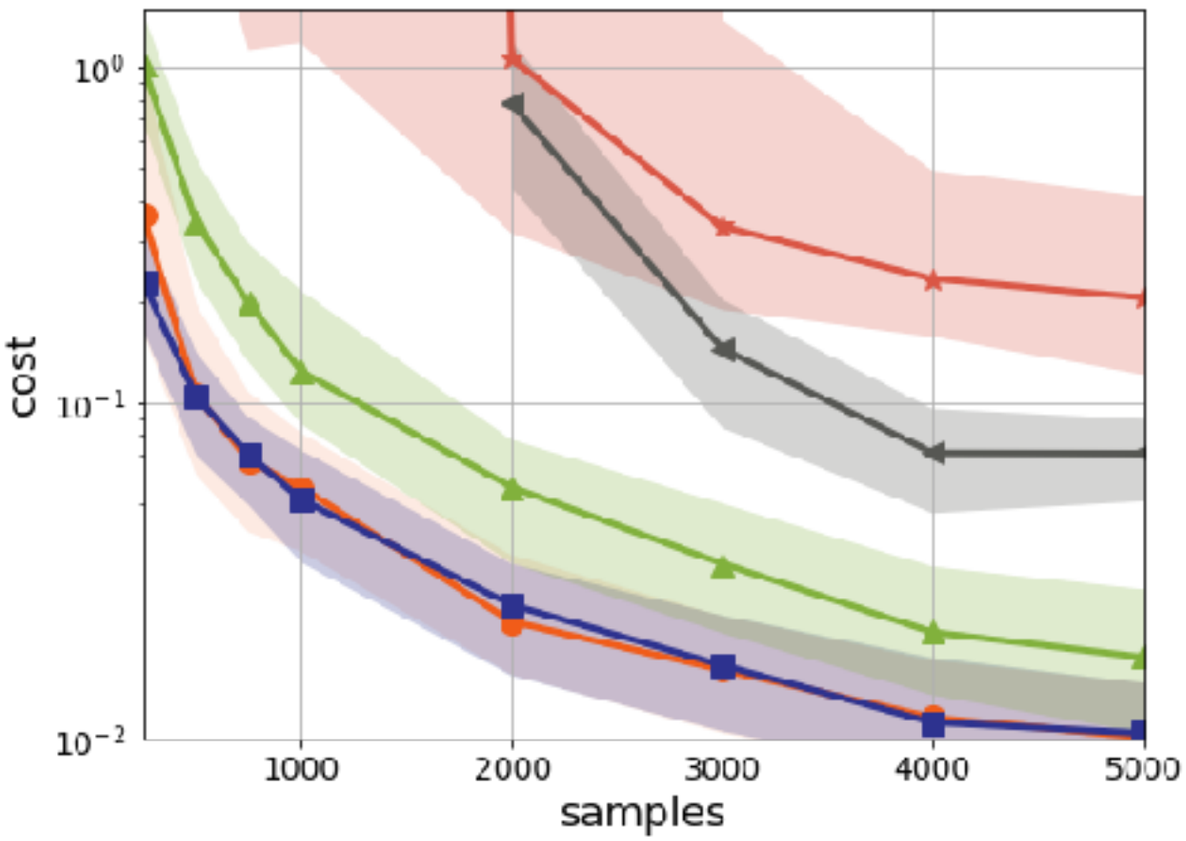
Slightly unstable system, system ID tends to think some nodes are stable

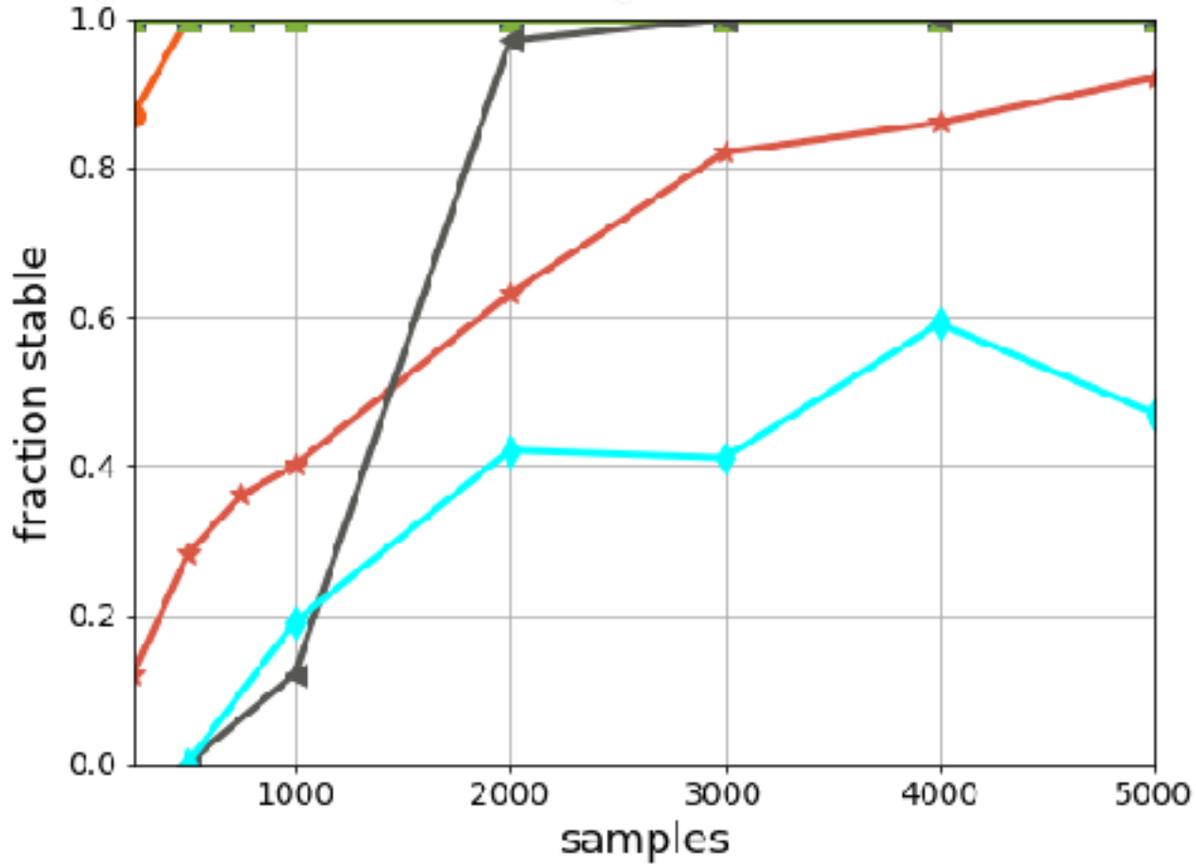




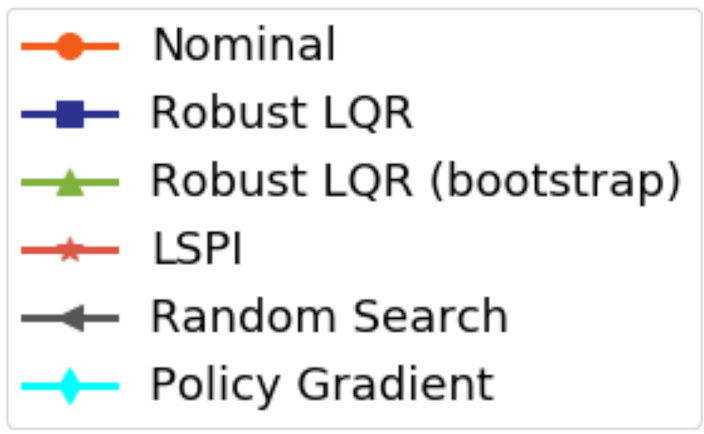
Least-squares estimate may yield unstable controller

Robust synthesis yields stable controller





Model-free performs worse than model-based



Even LQR is not simple!!!

$$\begin{aligned} \text{minimize} \quad & J := \sum_{t=1}^{\infty} x_t^T Q x_t + u_t^T R u_t \\ \text{s.t.} \quad & x_{t+1} = A x_t + B u_t + e_t \end{aligned}$$ *Gaussian noise*

- Coarse-ID control is the first non-asymptotic bound for LQR in this oracle model.
- The estimation results disagree with 50 papers on Cosma Shalizi's blog.
- The guarantees for least-squares estimation required some heavy machinery and are building on results from last year.
- The SDP relaxation uses brand new techniques in controller parameterization *(Systems Level Synthesis* by Matni *et al.)*
  - **Key insight:** Robustness makes analysis tractable

Even the simplest RL problems are really hard.

So far…

- Model based methods seem to perform better than model free ones in theory and practice.
- RL needs better baselines!
- Simple algorithms seem to be surprisingly competitive.
- Analysis of time series is annoyingly hard!

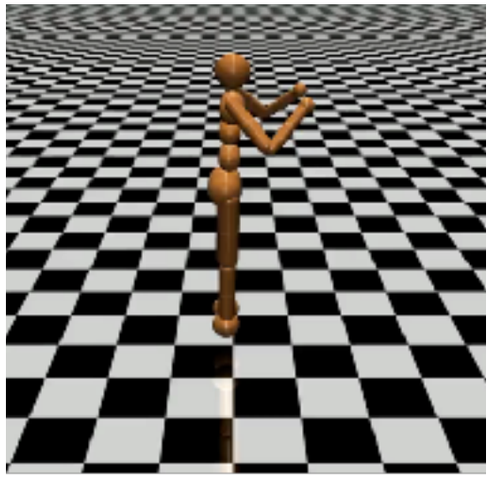
Extensions

- Poly-time Direct Adaptive Control (NeurIPS2018)
- Limitations of policy search (NeurIPS 2018)
- Safe exploration (submitted to ACC 2019)

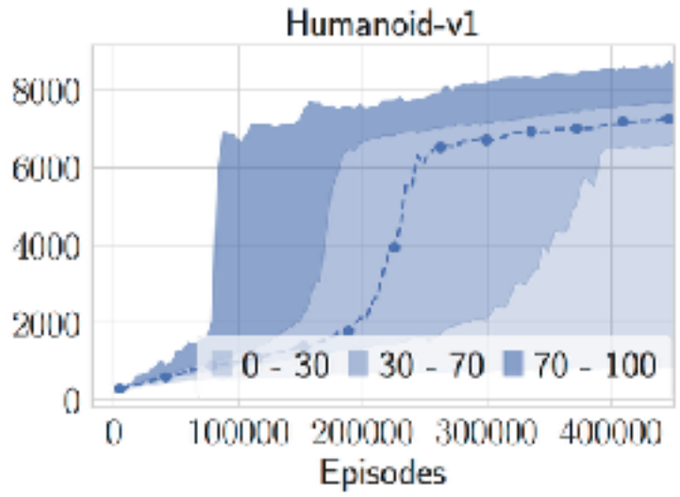
The Linearization Principle

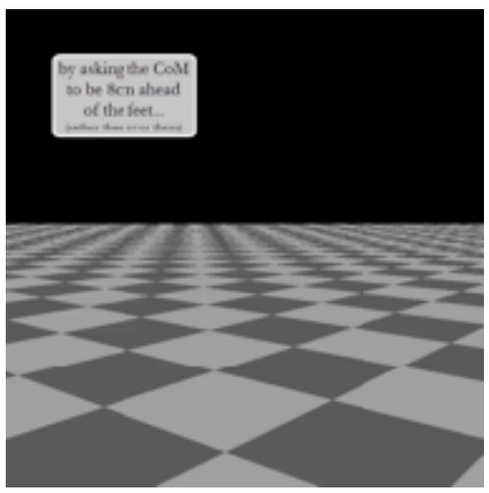
If a machine learning algorithm does crazy things when restricted to linear models, it's going to do crazy things on complex nonlinear models too.

What happens when we return to nonlinear models?



Simple, static, linear policies outperform deep RL and can be found with simple, standard optimization algorithms.





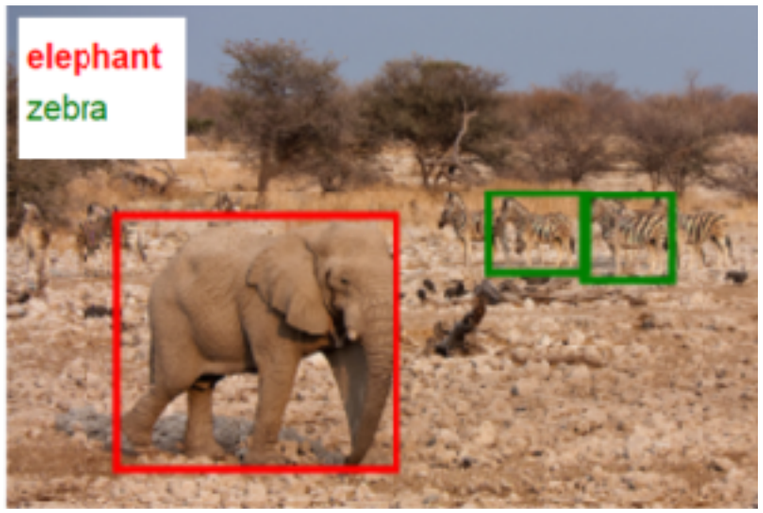
Model Predictive Control outperforms any direct policy search method.

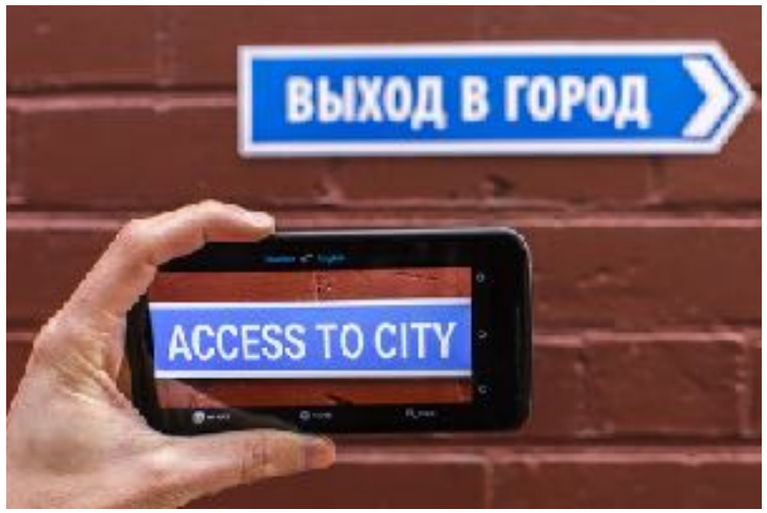
Video from Todorov Lab, 2012

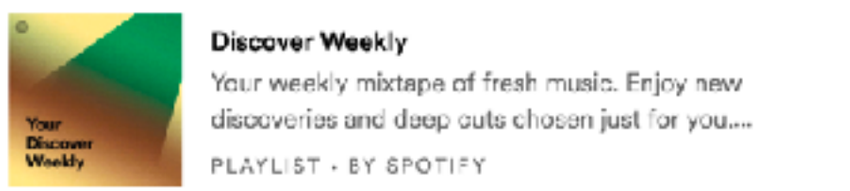
[Mania, Guy, R. 2018]

What is ML good for in control?

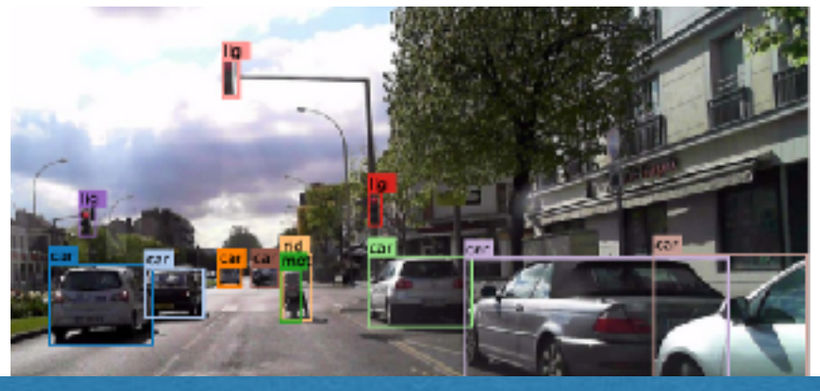
- Fundamentally, almost all machine learning successes are in *nonparametric prediction* (mostly classification).



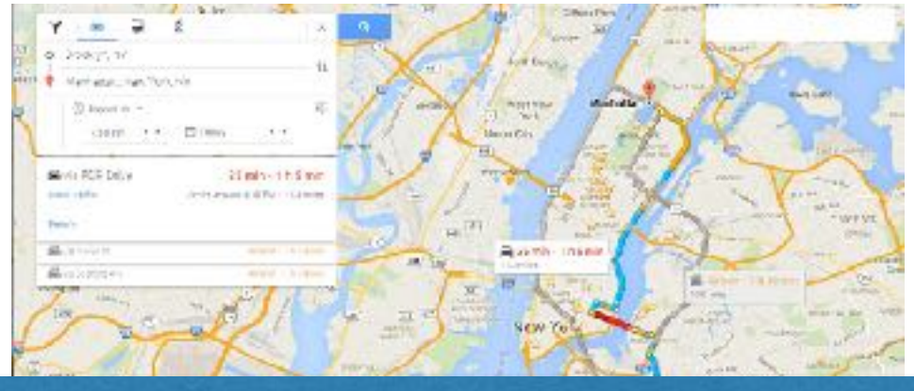




Perceptual sensors in the loop



Forecasting in MPC

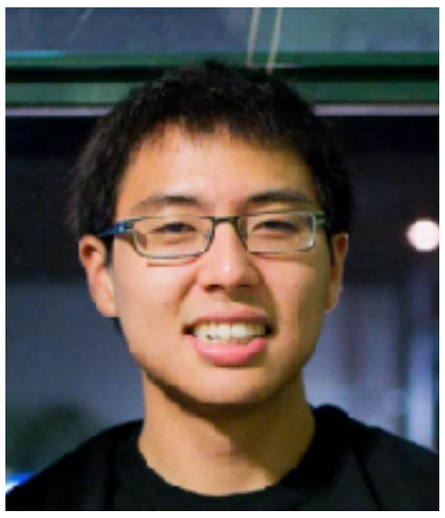


How to incorporate uncertain predictive perception in trustable, scalable, predictable autonomy?

Collaborators





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

References

- "On the Sample Complexity of the Linear Quadratic Regulator." S. Dean, H. Mania, N. Matni, B. Recht, and S. Tu. **arXiv:1710.01688**
- "Least-squares Temporal Differencing for the Linear Quadratic Regulator" S. Tu and B. Recht. In ICML 2018. **arXiv:1712.08642**
- "Learning without Mixing." H. Mania, M. I. Jordan, B. Recht, M. Simchowitz, and S. Tu. In COLT 2018. **arXiv:1802.08334**
- "Simple random search provides a competitive approach to reinforcement learning." H. Mania, A. Guy, and B. Recht. In NeurIPS 2018. **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. In NeurIPS 2018. **arXiv:1805.09388**
- "The Gap Between Model-Based and Model-Free Methods on the Linear Quadratic Regulator: An Asymptotic Viewpoint." S. Tu and B. Recht. **arXiv:1812.03565**
- A Tour of Reinforcement Learning: The View from Continuous Control." B. Recht. **arXiv:1806.09460**