reinforcement learning through the optimization lens

Benjamin Recht
University of California, Berkeley
trustable, scalable, predictable
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
Control Theory
continuous
model → action
IEEE Transactions

CS
Reinforcement Learning
discrete
data → action
Science Magazine
Today's talk will try to unify these camps and point out how to merge their perspectives.
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)
Control theory is the study of dynamical systems with inputs.

Reinforcement learning is discrete.

Simplest example: Partially Observed Markov Decision Process (POMDP)

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

\( x_t \) 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

- 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.

\[ x_{t+1} = Ax_t + Bu_t \]
\[ u_t = \pi_t(\tau_t) \]
Optimal control

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

s.t.
\[ x_{t+1} = f_t(x_t, u_t, e_t) \]
\[ u_t = \pi_t(\tau_t) \]

\( C_t \) 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.
Optimal control

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

Hydrokinetic (HVAC) room state and action:

\[
M \dot{T} = \dot{Q} + m_s c_p (T_s - T)
\]
Identify everything

Identify a coarse model

We don’t need no stinking models!

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

\[ M \dot{T} = \dot{Q} + \dot{m_s} c_p (T_s - T) \]

- 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

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

\( C_t \) 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

\[ Z_{t+1} = Z_t + v_t \]
\[ v_{t+1} = v_t + a_t \]
\[ m a_t = u_t \]

minimize \[ \sum_{t=0}^{T} 1_{ |(x_t)_1| > \epsilon } \]

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

\[ x_t = \begin{bmatrix} Z_t \\ v_t \end{bmatrix} \]
Newton's Laws

\[ z_{t+1} = z_t + v_t \]
\[ v_{t+1} = v_t + a_t \]
\[ ma_t = u_t \]

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

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

s.t. $x_{t+1} = A x_t + B u_t + e_t$

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

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

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

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

s.t. \( x_{t+1} = A x_t + B u_t + e_t \)

minimize \( \sum_{t=0}^{T} (x_t)^2 + ru_t^2 \)

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

s.t. \( x_{t+1} = A x_t + B u_t + e_t \)

Suppose \((A, B)\) unknown

What is the optimal estimation/design scheme?

How many samples are needed for near optimal control?
Optimal control

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

s.t.
\[
\begin{align*}
x_{t+1} &= f_t(x_t, u_t, e_t) \\
u_t &= \pi_t(\tau_t)
\end{align*}
\]

generic solutions with known dynamics:

Batch Optimization

Dynamic Programming
RL Methods

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

s.t. \[ x_{t+1} = f_t(x_t, u_t, e_t) \]
\[ u_t = \pi_t(\tau_t) \]

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

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

$u_t = \pi_t(\tau_t)$

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

Fit dynamics with \textit{supervised learning}:

$\hat{\varphi} = \arg\min_{\varphi} \sum_{t=0}^{N-1} \|x_{t+1} - \varphi(x_t, u_t)\|^2$

Solve approximate problem:

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

s.t. $x_{t+1} = \varphi(x_t, u_t) + \omega_t$

$u_t = \pi(\tau_t)$
“Simple” Example: LQR

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. \( x_{t+1} = A x_t + B u_t + e_t \)

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

Run an experiment for \( T \) steps with random input. Then

\[
\min_{(A,B)} \sum_{i=1}^{T} \| x_{i+1} - A x_i - B u_i \|^2
\]

If \( T \geq \tilde{O} \left( \frac{\sigma^2 (d + p)}{\lambda_{\min}(\Lambda_c) \epsilon^2} \right) \)

where \( \Lambda_c = A \Lambda_c A^* + B B^* \)

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

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

s.t.

\[ x_{t+1} = f_t(x_t, u_t, e_t) \]
\[ u_t = \pi_t(\tau_t) \]
Dynamic Programming

minimize \( \mathbb{E}_e \left[ \sum_{t=1}^{T} C_t(x_t, u_t) + C_f(x_{T+1}) \right] =: V_1(x) \)

s.t. \( x_{t+1} = f_t(x_t, u_t, e_t), \ x_1 = x \) \( u_t = \pi_t(\tau_t) \)

Terminal value:

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

Recursive formula (recurse backwards):

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

Optimal Policy:

\( \pi_k(\tau_k) = \arg \min_u C_k(x_k, u) + \mathbb{E}_e \left[ V_{k+1}(f_k(x_k, u, e)) \right] \)
```
“Simplest” Example: LQR

minimize $\mathbb{E} \left[ \sum_{t=1}^{T-1} x^*_t Q x_t + u^*_t R u_t + x^*_T P_T x_T \right]$

s.t. $x_{t+1} = A x_t + B u_t + e_t$

**Dynamic Programming:**

$V_T(x_T) = x^*_T P_T x_T$

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

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

$u_{T-1} = -(B^* P_T B + R)^{-1} B^* P_T A x_T =: K_{T-1} x_{T-1}$

$V_T(x_{T-1}) = x^*_{T-1} P_{T-1} x_{T-1}$

$P_{T-1} = Q + A^* P_T A - A^* P_T B \left( R + B^* P_T B \right)^{-1} B^* P_T A$
```
“Simplest” Example: LQR

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

s.t. \[ x_{t+1} = A x_t + B u_t + e_t \]

When \((A,B)\) known, optimal to build control \(u_t = K x_t\)

\[
u_t = -(B^* P B + R)^{-1} B^* P A x_t =: K x_t
\]

\[
P = Q + A^* P A - A^* P B (R + B^* P B)^{-1} B^* P A
\]

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

s.t.
\[
\begin{align*}
    x_{t+1} &= f_t(x_t, u_t, e_t) \\
    u_t &= \pi_t(\tau_t)
\end{align*}
\]

Approximate Dynamic Programming

Recursive formula:
\[
V_k(x) = \min_u C_k(x, u) + \mathbb{E}_e [V_{k+1}(f_k(x, u, e))]\]

Optimal Policy:
\[
\pi_k(\tau_k) = \arg\min_u C_k(x_k, u) + \mathbb{E}_e [V_{k+1}(f_k(x_k, u, e))]
\]
Approximate Dynamic Programming

Bellman Equation:

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

Optimal Policy:

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

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

Bellman Equation:

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

Optimal Policy:

\[
\pi(x) = \arg \min_u C(x, u) + \gamma \mathbb{E}_e [V_\gamma(f(x, u, e))] \\
\pi(x) = \arg \min_u Q(x, u)
\]
minimize  \[ \mathbb{E}_e \left[ \sum_{t=1}^{\infty} \gamma^t C(x_t, u_t) \right] \]

s.t.  
\[ x_{t+1} = f(x_t, u_t, e_t) \]
\[ u_t = \pi(\tau_t) \]

\[ Q(x, u) = C(x, u) + \mathbb{E}_e [\gamma \mathcal{V}_\gamma (f(x, u, e))] \]

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

Optimal Policy:
\[ \pi(x) = \arg \min_u Q(x, u) \]

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

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

Direct Policy Search
Sampling to Search

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

\[ \leq \min_{\theta} \mathbb{E}_{p(z; \theta)}[\Phi(z)] =: j(\theta) \]

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

\[ \nabla j(\theta) = \mathbb{E}_{p(z; \theta)}[\Phi(z) \nabla_{\theta} \log p(z; \theta)] \]
Reinforce Algorithm

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

\[ \nabla_\theta J(\theta) = \int \Phi(z) \nabla_\theta p(z; \theta) dz \]

\[ = \int \Phi(z) \left( \frac{\nabla_\theta p(z; \theta)}{p(z; \theta)} \right) p(z; \theta) dz \]

\[ = \int (\Phi(z) \nabla_\theta \log p(z; \theta)) p(z; \theta) dz \]

\[ = \mathbb{E}_{p(z; \theta)} [\Phi(z) \nabla_\theta \log p(z; \theta)] \]
Reinforce Algorithm

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

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

Sample \hspace{1cm} z_k \sim p(z; \vartheta_k)

Compute \hspace{1cm} G(z_k, \vartheta_k) = \Phi(z_k)\nabla_\vartheta \log p(z_k; \vartheta_k)

Update \hspace{1cm} \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:

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

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

Does this “solve” any discrete problem?
minimize $\mathbb{E}_e \left[ \sum_{t=1}^{T} C_t(x_t, u_t) \right]$

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

Direct Policy Search

Reinforce applied to either problems does not depend on the dynamics

Both are Derivative-free algorithms!
Policy Gradient

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

probabilistic policy

\[
G(\tau, \theta) = \left( \sum_{t=1}^{T} C(x_t, u_t) \right) \cdot \left( \sum_{t=0}^{T-1} \nabla_{\theta} \log p_{\theta}(u_t|x_t; \theta) \right)
\]
Policy Gradient for LQR

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

"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 \( u_t = K x_t + \nu_t : \quad \tau = \{x_1, \ldots, x_T\} \)

- Compute cost: \( C(\tau) = \sum_{t=1}^{T} x_t^* Q x_t + u_t^* R u_t \)

- Update: \( K_{\text{new}} \leftarrow K_{\text{old}} - \alpha_t \sum_{t=0}^{T-1} \nu_t x_t^* \)

Policy gradient only has access to 0-th order information!!!
Random Search

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

s.t.

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

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

parameter perturbation

Direct Policy Search

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

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

$\omega_i \sim \mathcal{N}(0, 1)$

random finite difference approximation to the gradient

aka… $(\mu, \lambda)$-Evolution Strategies

SPSA

Bandit Convex Opt
Random Search for LQR

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

“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: \[ K \leftarrow K - \alpha_t J(\tau) \nu \]
• 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

minimize \( \sum_{t=0}^{T} (x_t)^2 + ru_t^2 \)

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
“Simplest” Example: LQR

\[ \text{minimize} \quad \sum_{t=0}^{T} (x_t^2 + ru_t^2) \]

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?”

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.”

There has to be a better way!

[blog.openai.com/openai-baselines-dqn/](http://blog.openai.com/openai-baselines-dqn/)

[arxiv:1709.06560](https://arxiv.org/abs/1709.06560)
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)

\[
\begin{align*}
\text{minimize} \quad & u^T Q u \\
\text{subject to} \quad & x = Bu + x_0 \\
\end{align*}
\]

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

Estimate \( B \): \( \text{minimize} \quad \sum_{i=1}^{N} \|Bu_i + x_0 - x_i\|^2 \)

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

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

Robust optimization problem:

\[
\begin{align*}
\text{minimize} \quad & u^T \sup_{\|\Delta_B\| \leq \epsilon} \| Q^{1/2} (x - \Delta_B u) \| \\
\text{subject to} \quad & x = \hat{B}u + x_0 \\
\end{align*}
\]

B unknown!
Coarse-ID control (static case)

\[
\begin{align*}
\text{minimize} & \quad u^* Q x \\
\text{subject to} & \quad x = Bu + x_0
\end{align*}
\]

\(B\) unknown!

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

Estimate \(B\): \(\min_B \sum_{i=1}^N \|Bu_i + x_0 - x_i\|^2\) \(\hat{B}\)

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

Solve robust optimization problem:
\[
\begin{align*}
\text{minimize} & \quad \sup_{\|\Delta_B\| \leq \epsilon} \|Q^{1/2}(x - \Delta_B u)\| \\
\text{subject to} & \quad x = \hat{B}u + x_0
\end{align*}
\]

Relaxation: (Triangle inequality!)
\[
\begin{align*}
\text{minimize} & \quad \|Q^{1/2}x\| + \epsilon \lambda \|u\| \\
\text{subject to} & \quad x = \hat{B}u + x_0
\end{align*}
\]
Coarse-ID control (static case)

\[
\begin{align*}
\text{minimize} & \quad x^* Q x \\
\text{subject to} & \quad x = Bu + x_0
\end{align*}
\]

Collect data: \( \{(x_i, u_i)\} \)

Estimate \( B \): \( \text{minimize} \quad \sum_{i=1}^{N} \|Bu_i - x_i\|^2 \quad \rightarrow \quad \hat{B} \)

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

Relaxation:

- (Triangle inequality!)
- \( \text{minimize} \quad \|Q^{1/2}x\| + \epsilon \lambda \|u\| \)
- \( \text{subject to} \quad x = \hat{B}u + x_0 \)

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)

\[
\begin{align*}
\text{minimize} & \quad x^* Q x \\
\text{subject to} & \quad x = Bu + x_0
\end{align*}
\]

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

Estimate \( B \):
\[
\text{minimize}_B \quad \sum_{i=1}^{N} \|Bu_i + x_0 - x_i\|^2 \quad \hat{B}
\]

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

Relaxation: (Triangle inequality!)
\[
\begin{align*}
\text{minimize}_u & \quad \|Q^{1/2} x\| + \epsilon \lambda \|u\| \\
\text{subject to} & \quad x = \hat{B} u + x_0
\end{align*}
\]

Generalization bound
\[
\text{cost}(\hat{u}) = \text{cost}(u_*) + \mathcal{O}(\epsilon)
\]
Coarse-ID optimization

\[
\begin{align*}
\text{minimize} \quad & f(x) \\
\text{subject to} \quad & g(x; \vartheta) \leq 0
\end{align*}
\]

\[\vartheta \text{ unknown!}\]

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

Estimate \( \vartheta: \quad \hat{\vartheta} \)

Guarantee: \( \text{dist}(\vartheta, \hat{\vartheta}) \leq \epsilon \) with high probability

Relaxation:

\[
\begin{align*}
\text{minimize} \quad & \hat{f}_{\text{robust}}(x) \\
\text{subject to} \quad & g(x; \hat{\vartheta}) \leq 0
\end{align*}
\]

Generalization bound:

\( f(\hat{x}) \leq f(x_*) + \text{err}(\epsilon, \|f - \hat{f}_{\text{robust}}\|, x_*, \vartheta) \)
“Simple” Example: LQR

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. \( x_{t+1} = A x_t + B u_t + e_t \) \hspace{1cm} Gaussian noise

“Obvious strategy”: Estimate \((A, B)\), build control \( u_t = K x_t \)

Run an experiment for \( T \) steps with random input. Then

minimize \((A, B)\) \( \sum_{i=1}^{T} \| x_{i+1} - A x_i - B u_i \|^2 \)

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

\[
\min_u \quad \sup_{\|\Delta_A\|_2 \leq \epsilon_A, \|\Delta_B\|_2 \leq \epsilon_B} \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} x_t^* Qx_t + u_t^* R u_t \\
\text{s.t.} \quad x_{t+1} = (\hat{A} + \Delta_A)x_t + (\hat{B} + \Delta_B)u_t
\]

Solving an SDP relaxation of this robust control problem yields

\[
\frac{J(\hat{K}) - J^*}{J^*} \leq C \Gamma_{cl} \left( \lambda_{\min}(\Lambda_c)^{-1/2} + \|K^*\|_2 \right) \sqrt{\frac{\sigma^2(d+p)}{T}} \quad \text{w.h.p.}
\]

\[
\Lambda_c = AA^* + BB^* \quad \text{controllability Gramian}
\]

\[
\Gamma_{cl} := \|(zI - A - BK^*)^{-1}\|_{\mathcal{H}_\infty} \quad \text{closed loop gain}
\]

This also tells you when your cost is finite!

[Dean, Mania, Matni, R., Tu 2017]
Why robust?

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

\[
x_{t+1} = \begin{bmatrix} 1.0 & 0.0 & 0 \\ 0.0 & 1.0 & 0.0 \\ 0 & 0.0 & 1.0 \end{bmatrix} x_t + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} u_t + e_t
\]
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 $J := \sum_{t=1}^{\infty} x_t^T Q x_t + u_t^T R u_t$

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

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

where $\Lambda_c = AA_c A^* + BB^*$  \text{controllability Gramian}

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

\[
\text{minimize } \lim_{T \to \infty} \mathbb{E} \left[ \frac{1}{T} \sum_{t=1}^{T} x_t^* Q x_t + u_t^* R u_t \right]
\]
\[
\text{s.t. } x_{t+1} = A x_t + B u_t + e_t
\]

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**

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

<table>
<thead>
<tr>
<th>Task</th>
<th>RS</th>
<th>NG-lin</th>
<th>NG-rbf</th>
<th>TRPO-nn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swimmer-v1</td>
<td>365</td>
<td>366</td>
<td>365</td>
<td>131</td>
</tr>
<tr>
<td>Hopper-v1</td>
<td>3909</td>
<td>3651</td>
<td>3810</td>
<td>3668</td>
</tr>
<tr>
<td>HalfCheetah-v1</td>
<td>6722</td>
<td>4149</td>
<td>6620</td>
<td>4800</td>
</tr>
<tr>
<td>Walker</td>
<td>11389</td>
<td>5234</td>
<td>5867</td>
<td>5594</td>
</tr>
<tr>
<td>Ant</td>
<td>5146</td>
<td>4607</td>
<td>4816</td>
<td>5007</td>
</tr>
<tr>
<td>Humanoid</td>
<td>11600</td>
<td>6440</td>
<td>6849</td>
<td>6482</td>
</tr>
</tbody>
</table>
Larger is better
Model Predictive Control

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

s.t. \[ x_{t+1} = f_t(x_t, u_t, e_t), \quad x_1 = x \]
\[ u_t = \pi_t(\tau_t) \]

Optimal Policy:

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

MPC: use the policy at every time step

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

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

References from the Actionable Intelligence Lab

- argmin.net
- B. Recht “A short and biased tour of reinforcement learning.” Coming soon…

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