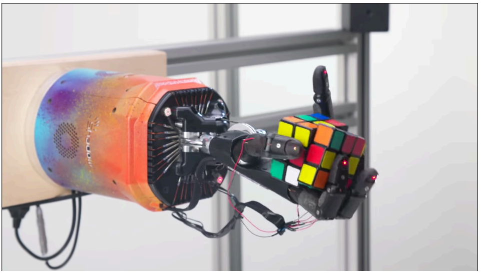
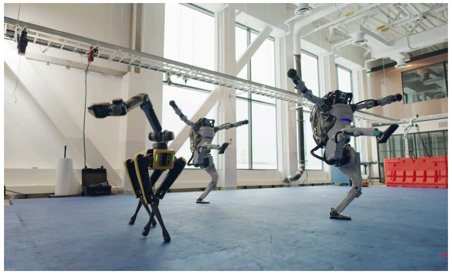
Reinforcement Learning & Optimal Control Overview

Yi Ma and Shankar Sastry

University of California, Berkeley

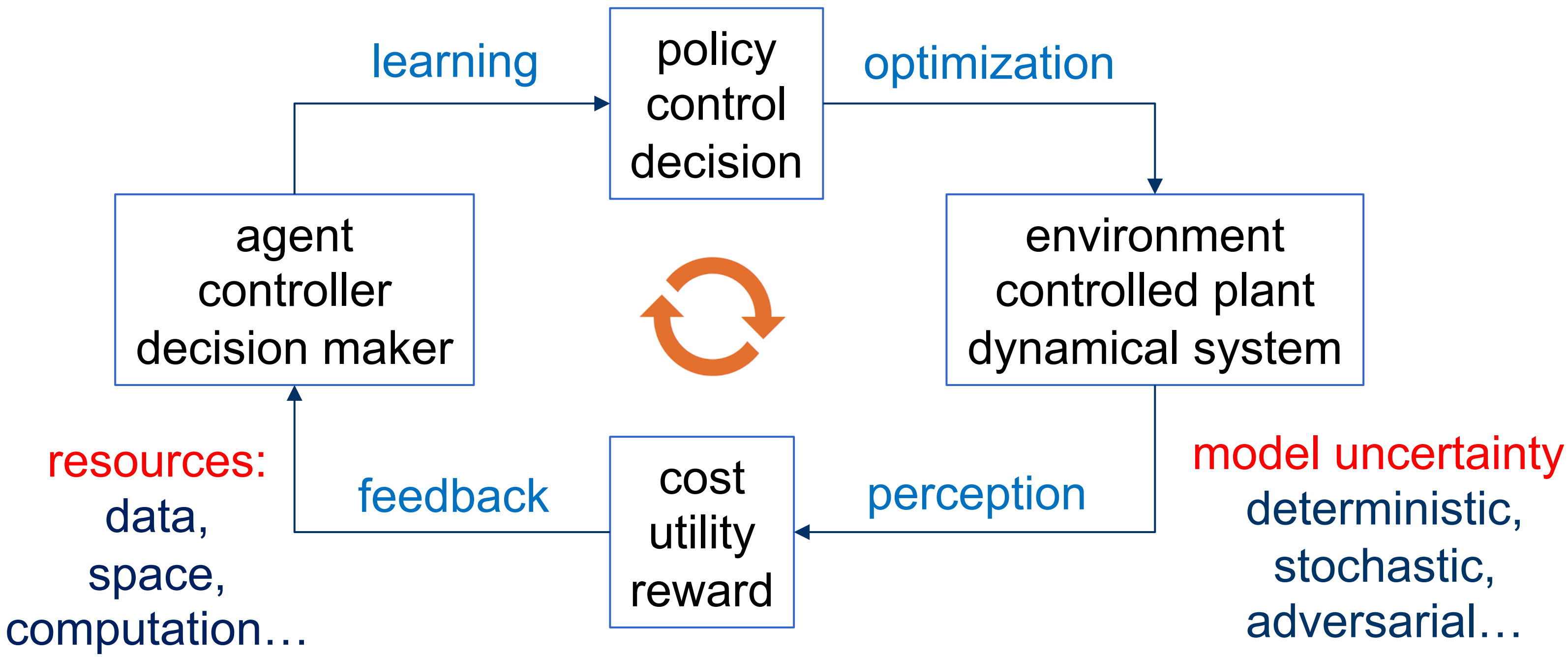




A Common Setting

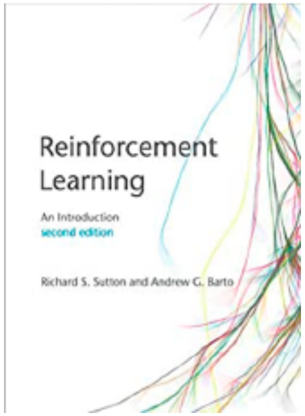
A Closed-Loop Autonomous System:

vacuuming robots, autonomous cars, video game players,
internet advertisements, trading stocks, animals in the wild…



A Brief (Recent) History

Evolution of Approximate DP/RL



AI/RL

Learning through Data/Experience

Simulation, Model-Free Methods

Feature-Based Representations

A*/Games/ Heuristics

Complementary Ideas

Late 80s-Early 90s

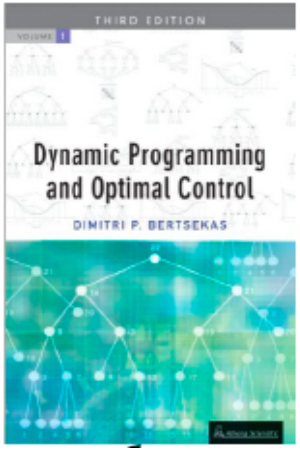
Decision/ Control/DP

Principle of Optimality

Markov Decision Problems

POMDP

Policy Iteration Value Iteration



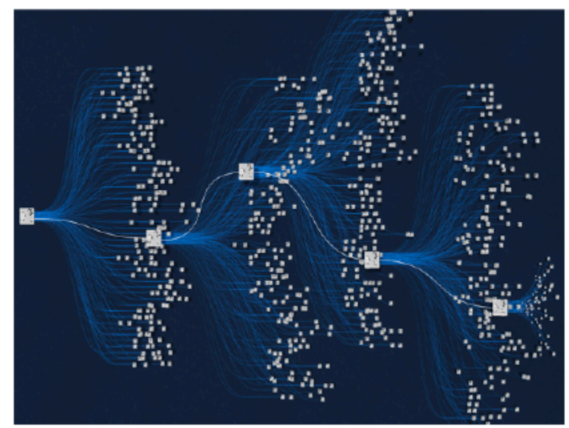
Historical highlights

- Exact DP, optimal control (Bellman, Shannon, and others 1950s ...)
- AI/RL and Decision/Control/DP ideas meet (late 80s-early 90s)
- First major successes: Backgammon programs (Tesauro, 1992, 1996)
- Algorithmic progress, analysis, applications, first books (mid 90s ...)
- Machine Learning, BIG Data, Robotics, Deep Neural Networks (mid 2000s ...)
- AlphaGo and AlphaZero (DeepMind, 2016, 2017)

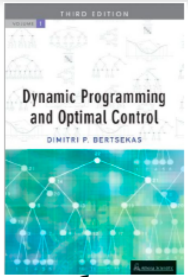
New Challenges

In RL, an agent learns by interacting with an environment

- unknown or changing environments
- delayed rewards or feedback
- enormous state and action space
- nonconvexity



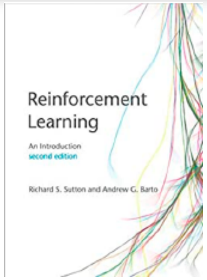
Terminology: State Space Model



OC/DP

environment
controlled plant
dynamical system

AI/RL



State and Control space: $\mathcal{S}, \mathcal{U}$

State: $x_k \in \mathcal{S}, k = 0, 1, \ldots$

Control: $u_k \in \mathcal{U}, k = 0, 1, \ldots$

Dynamical System:

$$x_{k+1} = f(x_k, u_k)$$

stochastic

$$x_{k+1} = f(x_k, u_k, w_k)$$

Output/observation (feature):

stochastic

$$y_k = h(x_k, u_k) + n_k$$

State and Action space: $\mathcal{S}, \mathcal{A}$

State: $s_t \in \mathcal{S}, t = 0, 1, \ldots$

Action: $a_t \in \mathcal{A}, t = 0, 1, \ldots$

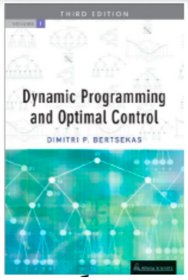
MDP Transition (or simulation):

$$\mathcal{T}_{ijk} = p(s_{t+1} = i \mid s_t = j, a_t = k)$$

Observation (feature):

$$p(o_t \mid s_t)$$

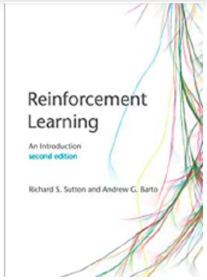
Terminology: Optimization Objective



OC/DP

policy
control
decision

AI/RL



Cost: $g(x_k, u_k) \in \mathbb{R}$

Total cost function:

$$J(x_0; u_0, \ldots, u_N) = \sum_{k=0}^{N} g(x_k, u_k)$$

Control law: $u(x_k)$; $u^*(x_k)$

$$x_{k+1} = f(x_k, u(x_k))$$
$$u_{k+1} = u(x_{k+1})$$

Value function (minimal cost to go):

$$J^*(x_0) = \min_{u(\cdot)} \sum_{k=0}^{N} g(x_k, u(x_k, k))$$

Reward: $r(s_t, a_t) \in \mathbb{R}$

Total reward (return):

$$J(s_1; a_1, \ldots, a_T) = \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[r(s_t, a_t)]$$

Policy: $\pi(a_t \mid s_t)$; $\pi^\star(a_t \mid s_t)$

$$p((s_{t+1}, a_{t+1}) \mid (s_t, a_t)) =$$
$$p(s_{t+1} \mid s_t, a_t)\pi(a_{t+1} \mid s_{t+1})$$

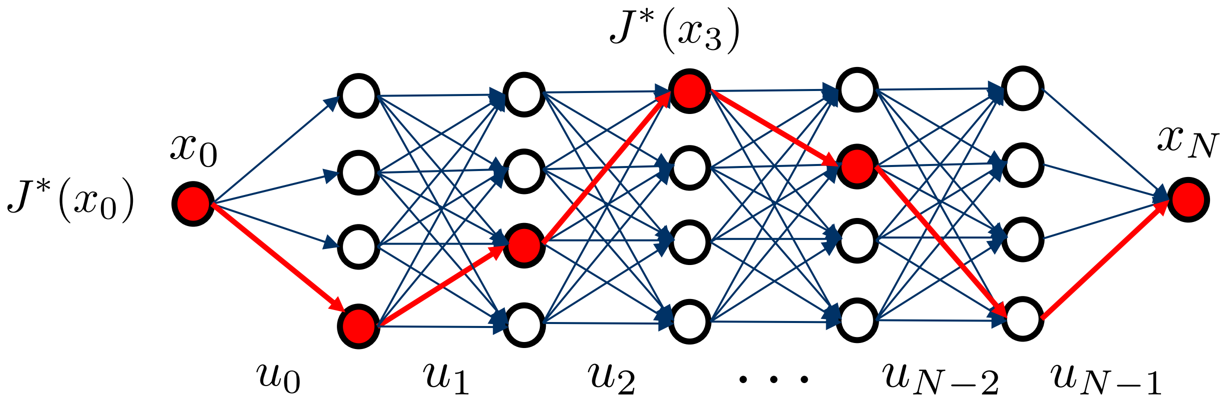
Value function (maximal return):

$$V^\star(s_1) = \max_{\pi(\cdot)} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_\pi[r(s_t, a_t)]$$

Principle of (Path) Optimality

Dido of Carthage…, Euler, Lagrange, Newton, Hamilton, Jacobi, Pontryagin, Bellman, Ford, Kalman

850 BC 1960 AC



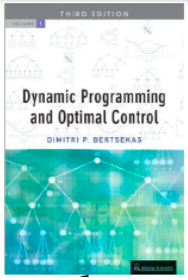
Principle of Optimality (***Richard Bellman'54***):
An optimal path has the property that any subsequent portion is optimal.

Dynamical Programming: A "Fixed-Point" Type Algorithm

$$J^*(x_k) = \min_{u_k} \left[g(x_k, u_k) + J^*(\underbrace{f(x_k, u_k)}_{x_{k+1}}) \right], \ \forall x_k$$

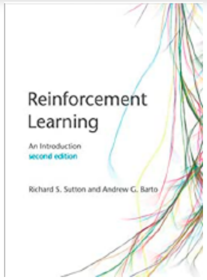
Bellman Operator: $\mathcal{T}(J^*) = J^*$ with $\mathcal{T}(J)(x) = \min_{u} \left[g(x,u) + J(f(x,u)) \right]$

Value Function versus Q-Function



OC/DP

AI/RL



Value function and Q-function:

$$J^*(x_k) = \min_{u_k} \underbrace{\left[g(x_k, u_k) + J^*(f(x_k, u_k))\right]}_{Q(x_k, u_k)}, \; \forall x_k$$

-- model free

$$V^\star(s_t) = \max_{\pi} \mathbb{E}_\pi \Big[\underbrace{p(s_{t+1} \mid s_t, a_t)}_{\mathcal{T}} \underbrace{\left[r(s_t, a_t) + V^\star(s_{t+1})\right]}_{Q(s_t, a_t)} \Big], \; \forall s_t$$

(many, many, many different ways to learn and solve them, depending on…)

Given the value or Q-function, the optimal control/policy and path:

$$u_k^* = \arg\min_{u_k} \underbrace{\left[g(x_k^*, u_k) + J^*(f(x_k^*, u_k))\right]}_{Q(x_k^*, u_k)},$$

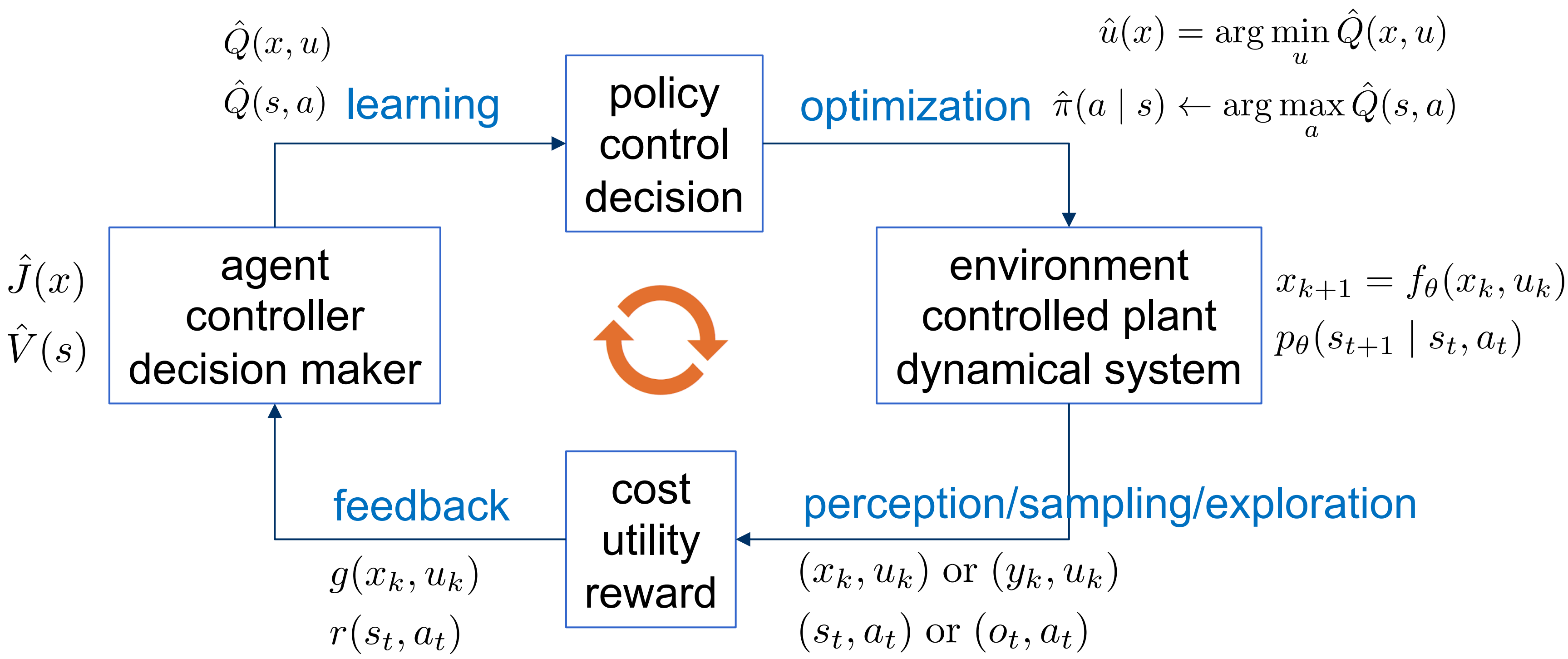
$$x_{k+1}^* = f(x_k^*, u_k^*)$$

$$\pi^\star(s_t) = \arg\max_{a_t} Q(s_t, a_t)$$

$$p(s_{t+1} \mid s_t, \pi^\star(s_t))$$

*In practice, states can be replaced by observations or "**features**" to relate to control or action.*

The Closed-Loop (Autonomous) System: Formal



From Principle to Computation!

What to Compute, and How?

	OC/DP	AI/RL
Optimal value function:	$J^*(x)$,	$V^\star(s)$
Optimal Q-function:	$Q^*(x,u)$,	$Q^\star(s,a)$
Optimal control/policy:	$u^*(x)$,	$\pi^\star(a \mid s)$ (or $u^\star(y),\ \pi^\star(a \mid o)$)
System/model identification:	$f^*(x,u)$,	$p^\star(s_{t+1} \mid s_t, a_t)$

Closed-form versus numerical solution (simulation & optimization)

LQR: $J^*(x_k) = \min_{u_k} \left[x_k^T Q x_k + u_k^T R u_k + J^*(A x_{k+1} + B u_k) \right]$

The Riccati equation (Kalman Filter '60):

$$K_k = -(\bar{R} + \bar{B}^\intercal V_{k+1} \bar{B})^{-1} \bar{B}^\intercal V_{k+1} \bar{A} \tag{48}$$

$$V_k = \bar{Q} + \bar{A}^\intercal V_{k+1} \bar{A} - \bar{A}^\intercal V_{k+1} \bar{B} (\bar{R} + \bar{B}^\intercal V_{k+1} \bar{B})^{-1} \bar{B}^\intercal V_{k+1} \bar{A} \tag{49}$$

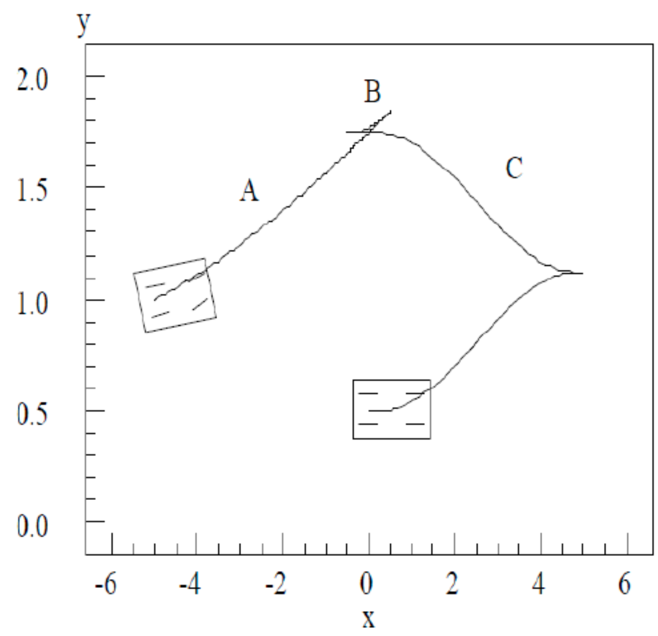
How to Compute?

Another Example:

Parallel Parking a (Nonholonomic) Car

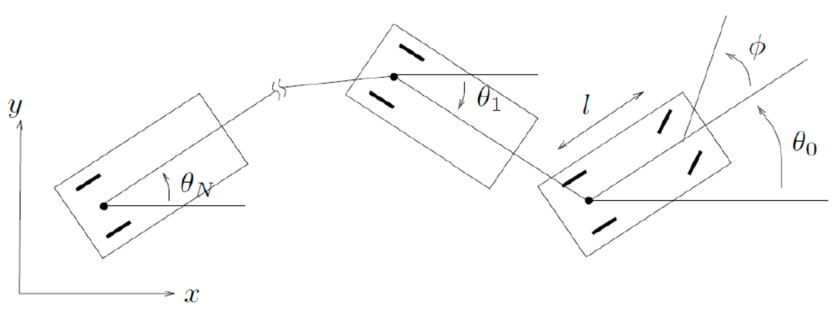
$$\begin{aligned}\dot{x} &= \cos\theta\, u_1\\ \dot{y} &= \sin\theta\, u_1\\ \dot{\theta} &= \frac{1}{l}\tan\phi\, u_1\\ \dot{\phi} &= u_2\end{aligned}$$

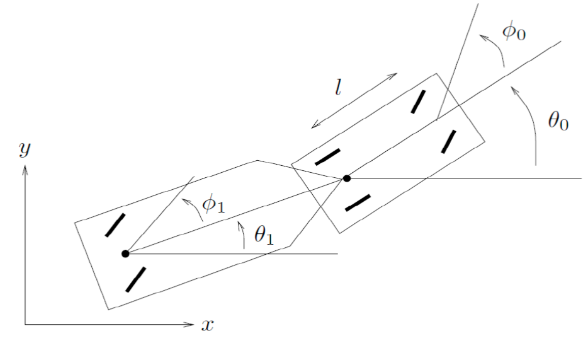
$$\min \int_0^1 ||u(t)||^2 dt$$



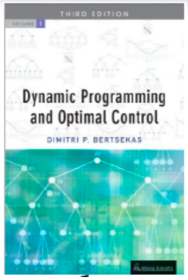
Optimal trajectories: *zig-zagging sinusoids (Brockett, Murray & Sastry'93,…)*

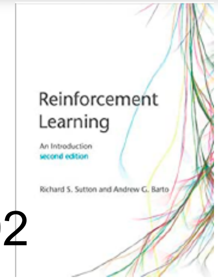
More Examples: chained form or *Goursat* normal form systems





Control versus Learning





OC/DP

- LQR
- Parallel parking
- Chained form systems
- Mechanical systems…

Conditions & Assumptions

- ○ clear model class/uncertainty
- ○ clear cost function
- ○ low to moderate dimension
- ○ continuous state/time…

AI/RL

- Backgammon: Tesauro, 1992
- Chess: Deep Blue, 1997
- Go: Alpha Go, 2017
- Video games, robots…

Conditions & Assumptions

- ○ unknown models (but can sample)
- ○ uncertain, long-horizon return
- ○ large-scale, high-dimensional
- ○ discrete state/time…

Solutions that work for a broad class of problems v.s. a few (important) instances

From Principle to Computation (Approximation)!

How to COMPUTE if no analytic or closed-form solution?

Major Approaches to Compute the Approximate Cost Function $\tilde{J}$

Problem approximation

Use as $\tilde{J}$ the optimal cost function of a related problem (computed by exact DP)

Rollout and model predictive control

Use as $\tilde{J}$ the cost function of some policy (computed somehow, perhaps according to some simplified optimization process)

Use of neural networks and other feature-based architectures

They serve as function approximators (usually obtained through off-line training)

Use of simulation to generate data to “train" the architectures

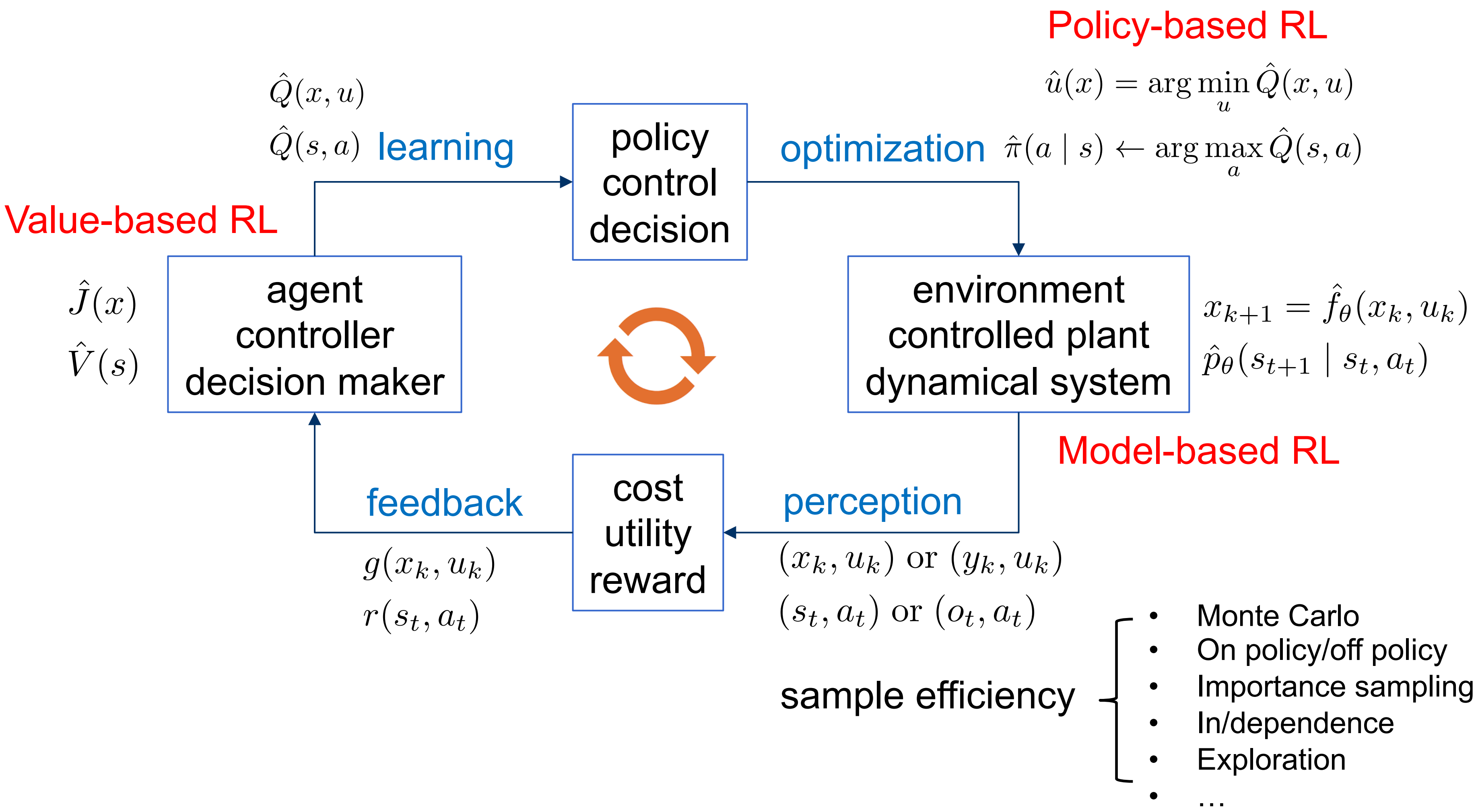
Approximation architectures involve parameters that are “optimized" using data

Policy iteration/self-learning, repeated policy changes

Multiple policies are sequentially generated; each is used to provide the data to train the next

What to Learn or Compute?

A Closed-Loop Autonomous System:



From Principle to Computation: Scalability

How to COMPUTE?

Chess (Deep Blue, 1997)

35^{80}

$|\mathcal{A}| \approx 35 \quad |T| = 80$

Go (Alpha Go, 2017)

250^{150}

$|\mathcal{A}| \approx 250 \quad |T| = 150$

\# of atoms in the universe

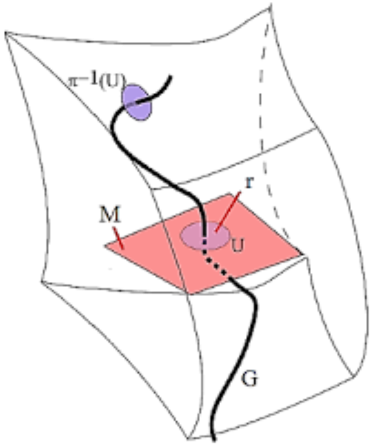
10^{82}

Unfortunately, there is no closed-form solution...
How to avoid the "**curse of dimensionality**" at all?

The solutions (functions) have low-dimensional structure!

$$\pi^\star(a \mid s) \approx \hat{\pi}(a; h_1(s,\theta), \ldots, h_d(s,\theta)), \quad h_i(s,\theta) \in \mathbb{R}$$

$$V^\star(s) \approx \hat{V}(h_1(s,\theta), \ldots, h_d(s,\theta)), \quad h_j(s,\theta) \in \mathbb{R}$$



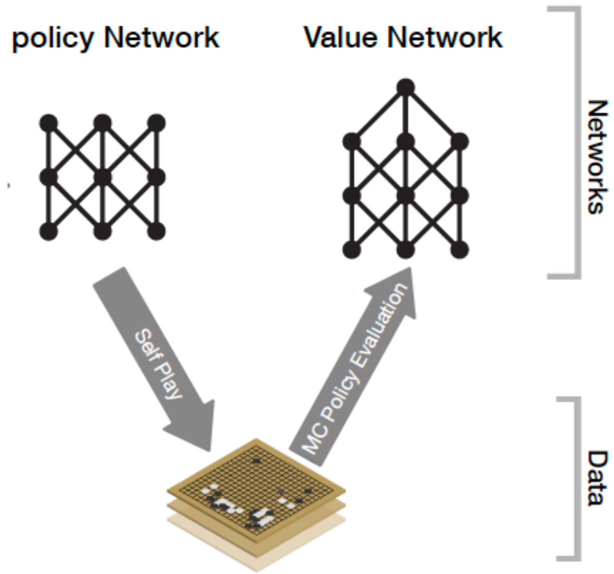
How to Approximate?

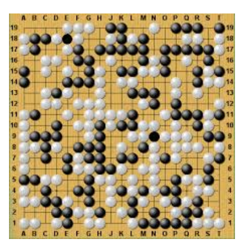
Policy: $\pi^\star(a \mid s) \approx \hat{\pi}(a; f_1(s,\theta), \ldots, f_d(s,\theta)), \quad f_i(s,\theta) \in \mathbb{R}$

Value: $V^\star(s) \approx \hat{V}(f_1(s,\theta), \ldots, f_d(s,\theta)), \quad f_j(s,\theta) \in \mathbb{R}$

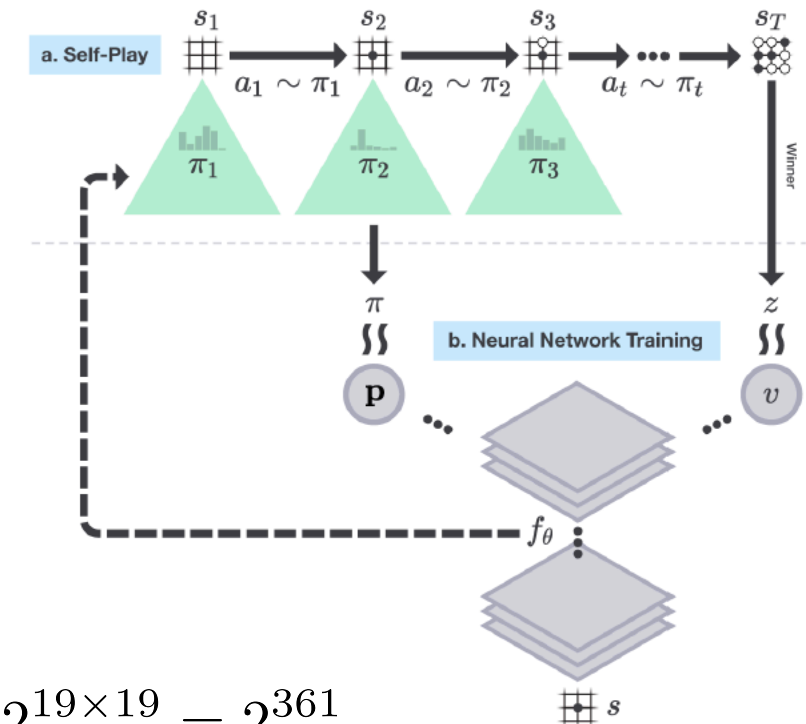
Nonlinear & sparse regression!

Alpha Go, 2017





$$2^{19\times 19} = 2^{361}$$



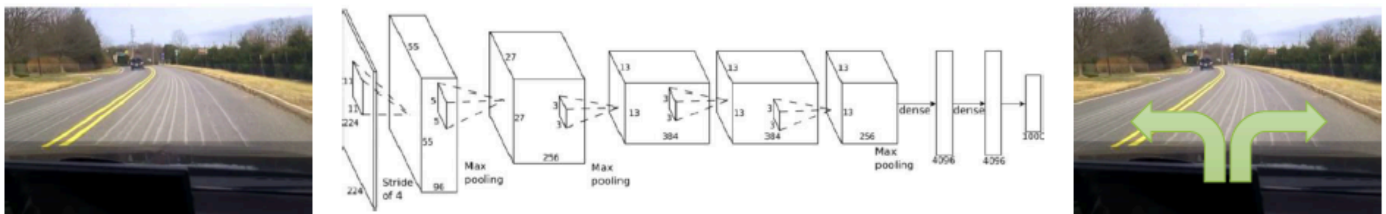
How to Approximate?

Policy: $\pi^*(a \mid o) \approx \hat{\pi}(a; h_1(o,\theta), \ldots, h_d(o,\theta)), \quad h_i(o,\theta) \in \mathbb{R}$

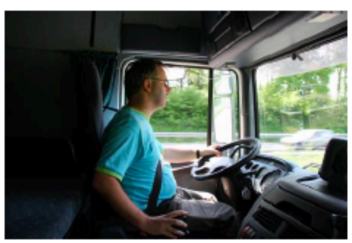
Value: $V^*(o) \approx \hat{V}(h_1(o,\theta), \ldots, h_d(o,\theta)), \quad h_j(o,\theta) \in \mathbb{R}$

Nonlinear & sparse regression!

Autonomous Driving



$\mathbf{o}_t$ $\quad \pi_\theta(\mathbf{a}_t|\mathbf{o}_t) \quad$ $\mathbf{a}_t$



$\mathbf{o}_t$

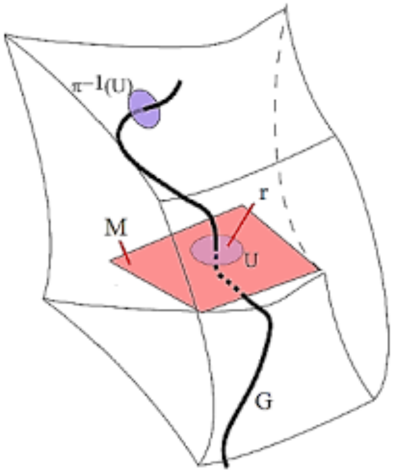
$\mathbf{a}_t$

$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

How to Learn Low-Dimensional Structures

In computer vision, we have been dealing with high-dimensional data with low-dimensional structures all the time!

Figure 4: Examples of rotated images of MNIST digits, each rotated by 18°. (**Left**) Diagram for polar coordinate representation; (**Right**) Rotated images of digit '0' and digit '1'.



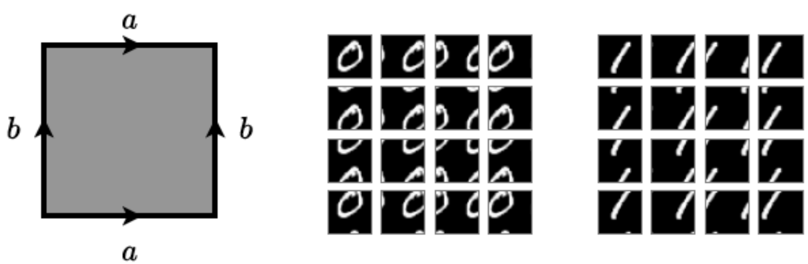
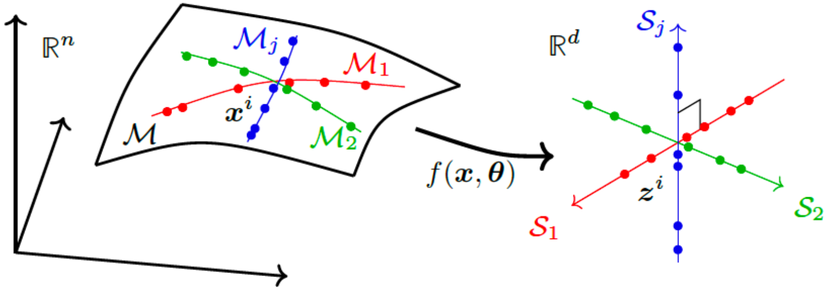


Figure 5: Examples of translated images of MNIST digits (with stride=7). (**Left**) A torus on which 2D cyclic translation is defined; (**Right**) Cyclic translated images of digit '0' and digit '1'.

Goal & role of deep networks:

- **Compression**
- **Optimization**
- **Linearization**



Some Representative Algorithms – Value-Based RL

Bellman Operator: $\mathcal{T}(Q)(s,a) \doteq r(s,a) + \mathbb{E}_{p(s'|s,a)}[\max_{a'} Q(s',a')]$

Bellman Equation: $Q^\star$ is the unique "fixed point" to

$$\mathcal{T}(Q^\star) = Q^\star$$

Q-Learning (Chris Watkins & Peter Dayan 1992)

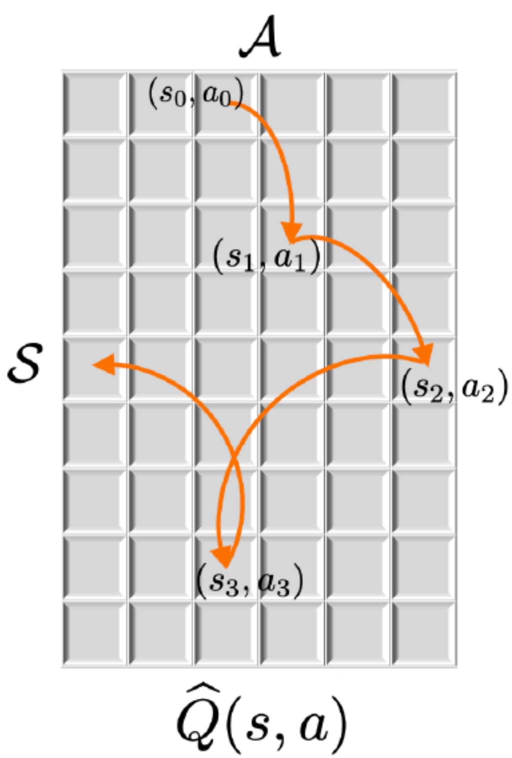
$$Q_{t+1}(s_t,a_t) = (1-\eta)\cdot Q_t(s_t,a_t) + \eta \cdot \mathcal{T}_t(Q_t)(s_t,a_t)$$

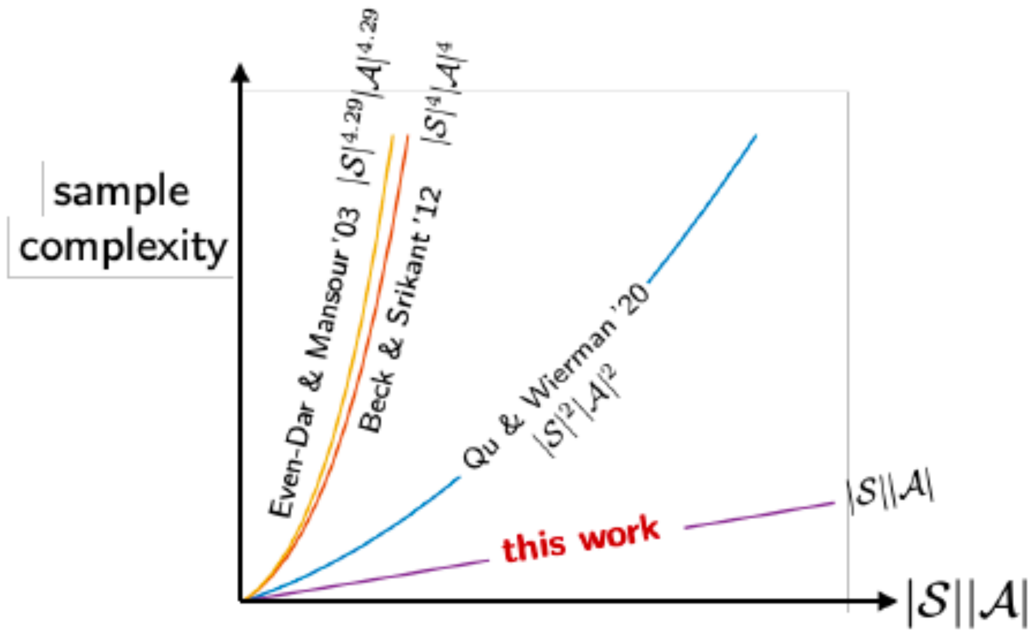
Model-free, stochastic approximation to solve the Bellman equation, updating only the (s_t,a_t) entry one at a time.

In practice, approximate value with a deep network and approximate "gradients", with many tricks.

Some Representative Algorithms – Value-Based RL

Question: how many samples are needed to ensure $\|\widehat{Q} - Q^\star\|_\infty \le \varepsilon$?



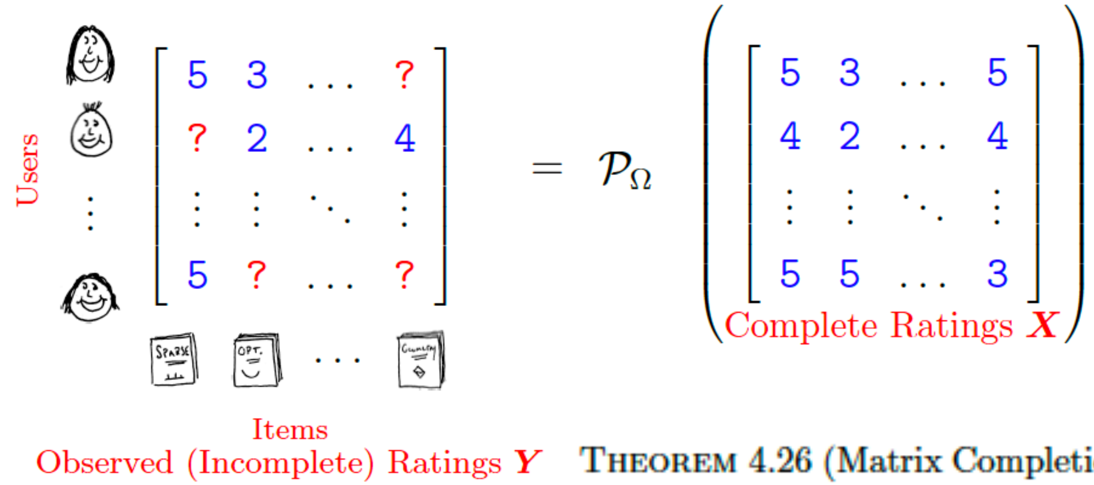


[Yuxin Chen et. al. 2021]

Yet, for Alpha Go $|\mathcal{S}| = 2^{361}$

Some Representative Algorithms – Value-Based RL

Low-rank Matrix Completion:



Theorem 4.26 (Matrix Completion via Nuclear Norm Minimization). *Let $\boldsymbol{X}_o \in \mathbb{R}^{n \times n}$ be a rank-r matrix with incoherence parameter ν. Suppose that we observe $\boldsymbol{Y} = \mathcal{P}_\Omega[\boldsymbol{X}_o]$, with Ω sampled according to the Bernoulli model with probability*

$$p \geq C_1 \frac{\nu r \log^2(n)}{n}. \tag{4.4.18}$$

Then with probability at least $1 - C_2 n^{-c_3}$, $\boldsymbol{X}_o$ is the unique optimal solution to

$$\text{minimize} \quad \|\boldsymbol{X}\|_* \quad \text{subject to} \quad \mathcal{P}_\Omega[\boldsymbol{X}] = \boldsymbol{Y}. \tag{4.4.19}$$

The real "matrix" $\hat{Q}(s, a)$ has low-dimensional structure!

Harnessing Structures for Value-based Planning and Reinforcement Learning, Yuzhe Yang, Guo Zhang, Zhi Xu, Dina Katabi, ICLR 2020. (MIT)

High-Dim Data Analysis with Low-Dim Models, Wright and Ma, Cambridge Univ. Press, 2021

Some Representative Algorithms - Policy-Based RL

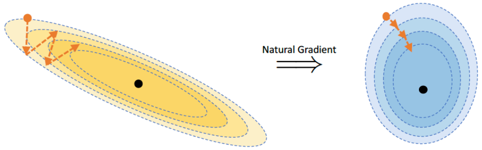
$$V^\star(s_1) = \max_{\pi(\cdot)} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_\pi[r(s_t, a_t)]$$

Policy Gradient Methods (Richard Sutton'00)

$$\max_\theta V^{\pi(\theta)}(s) = \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{\pi(\theta)}(s) \quad \text{with} \quad \pi(a \mid s, \theta) \doteq \frac{e^{h(s,a,\theta)}}{\sum_{a'} e^{h(s,a',\theta)}}$$

$$\theta_{k+1} = \theta_k + \eta \cdot \nabla_\theta V^{\pi(\theta_k)}(s)$$

Natural Policy Gradient (Kakate'02)



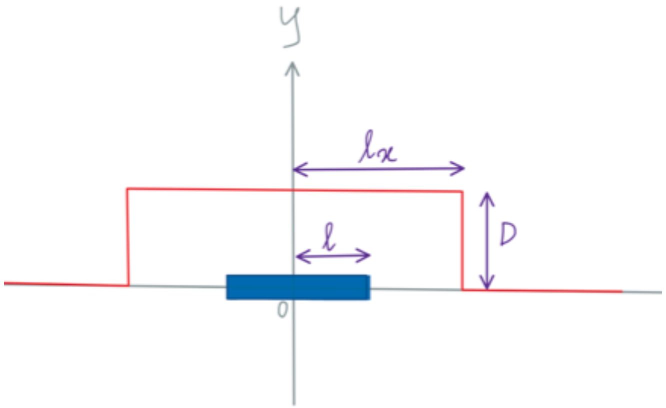
$$\theta_{k+1} = \theta_k + \eta \cdot (F(\theta_k))^\dagger \nabla_\theta V^{\pi(\theta_k)}(s)$$

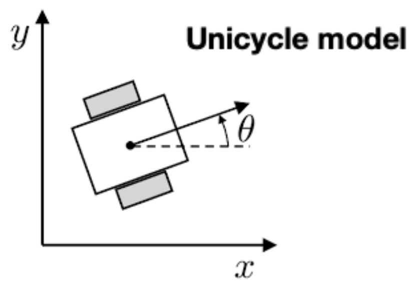
Fisher info. matrix

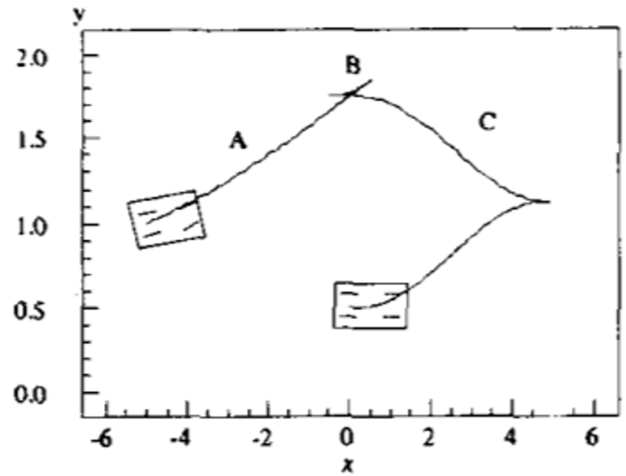
[Cen et. al. '20] *For any* $0 < \eta \leq (1-\gamma)/\tau$, *entropy-regularized NPG achieves*

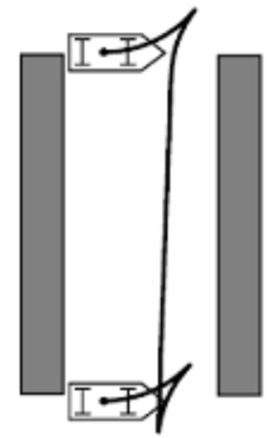
$$\|Q_\tau^\star - Q_\tau^{(t+1)}\|_\infty \leq C_1 \gamma (1 - \eta\tau)^t, \quad t = 0, 1, \cdots$$

Case Study of a Model Problem: Parallel Parking



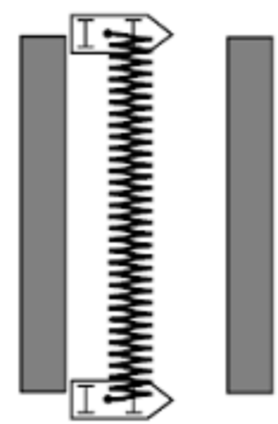




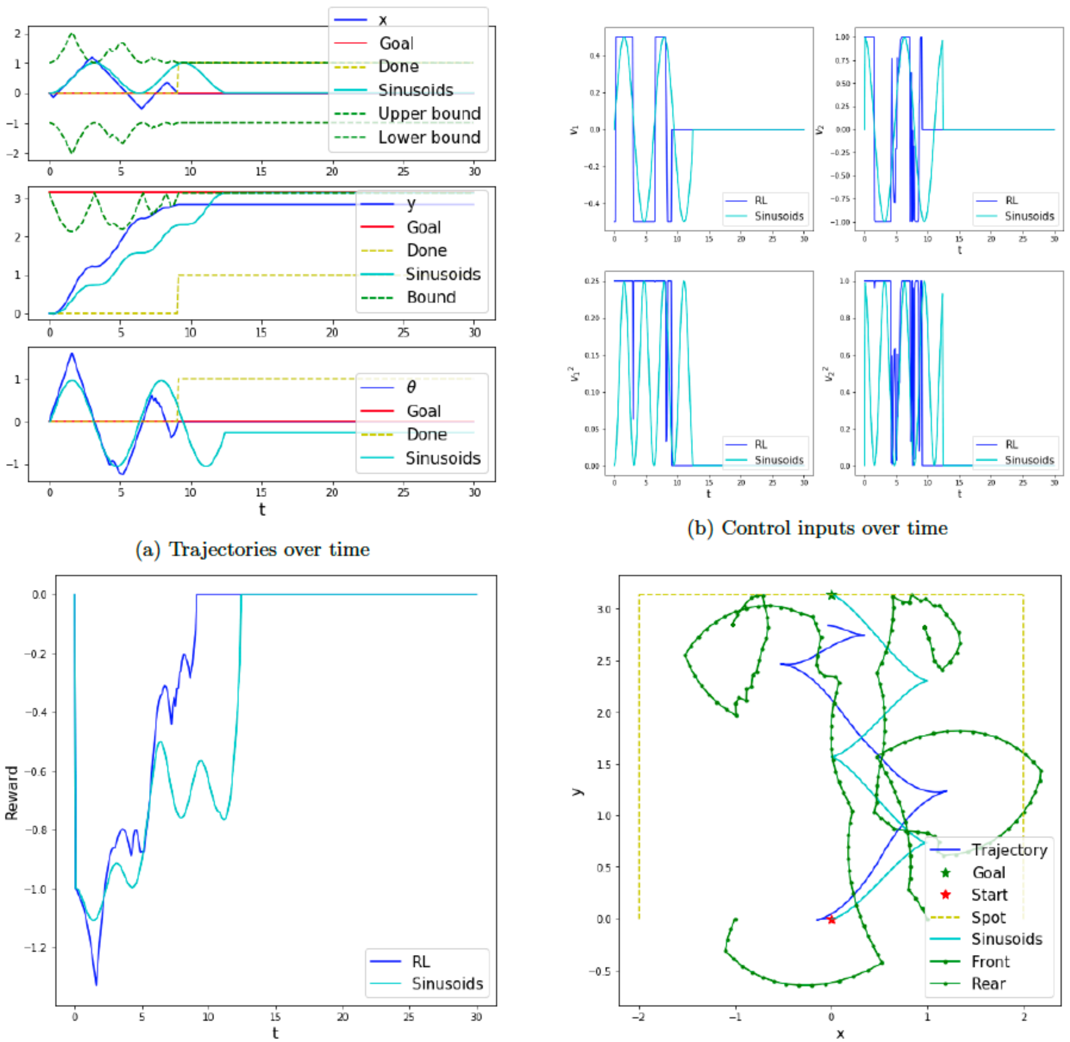


$$\begin{cases} \dot{x} = u_1 \cos\theta \\ \dot{y} = u_1 \sin\theta \\ \dot{\theta} = u_2 \end{cases} \Leftrightarrow \begin{cases} \dot{x} = v_1 \\ \dot{\theta} = v_2 \\ \dot{y} = v_1 \theta \end{cases}$$

What if we apply deep policy gradient?



Parallel Parking: Qualitative



(a) Trajectories over time

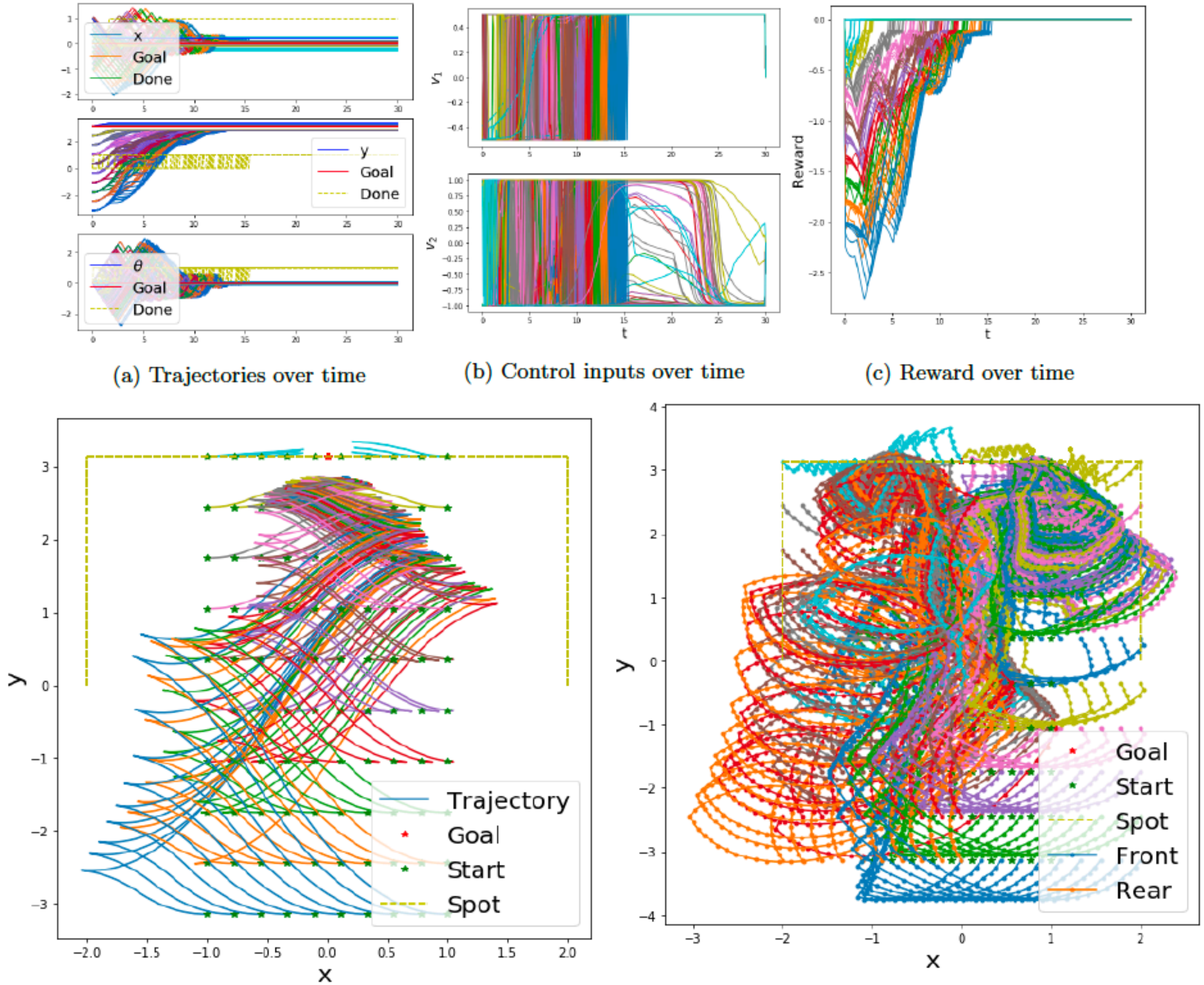
(b) Control inputs over time

(c) Reward over time

(d) Top down view of maneuver

Figure 4: $r(s_k, a_k) = -\|s_k - s_g\|$ - Minimizing distance to goal (run for 1000 epochs)

Parallel Parking: Qualitative



(a) Trajectories over time

(b) Control inputs over time

(c) Reward over time

(d) Top down view of maneuver

(e) Top down view of maneuver - Showing the front and rear of the car for inspection of safety

Figure 6: Testing for different initial conditions when trained using $r(s_k, a_k) = -\|s_k - s_g\|$ - Minimizing distance to goal

Parallel Parking: Quantitative

$$\min \int_0^1 ||u(t)||^2 dt$$

	RL	Sinusoids
$\int_0^T v_1^2(t)dt$	2.2146	1.5700
$\int_0^T v_2^2(t)dt$	7.5591	6.0200
$\int_0^T v_1^2(t)dt + \int_0^T v_2^2(t)dt$	9.7737	7.5900

Table 1: Comparison of energy consumed by the two methods

Takeaway messages about RL for parallel parking:

1. Higher cost (economy)
2. Very jittery control (comfort)
3. Do not always respect constraints (safety)
4. Hard to ensure accuracy in end position (precision)

Pause and Reflect

What about computational efficiency?
For LQR: Simple Random Search Provides a Competitive Approach to Reinforcement Learning, Horia Mania Aurelia Guy Benjamin Recht, 2018.

Empirically observed efficiency of RL does not come from the value-based methods or any smart sample schemes, it comes from exploiting the low-dimensionality of the solutions of the instances!

Diligence (by machine) is not intelligence!

What are other techniques that are effective in dealing with structured complexities in problems and enhance computation efficiency?

Real-World Robotic System Design: Quadrupedal

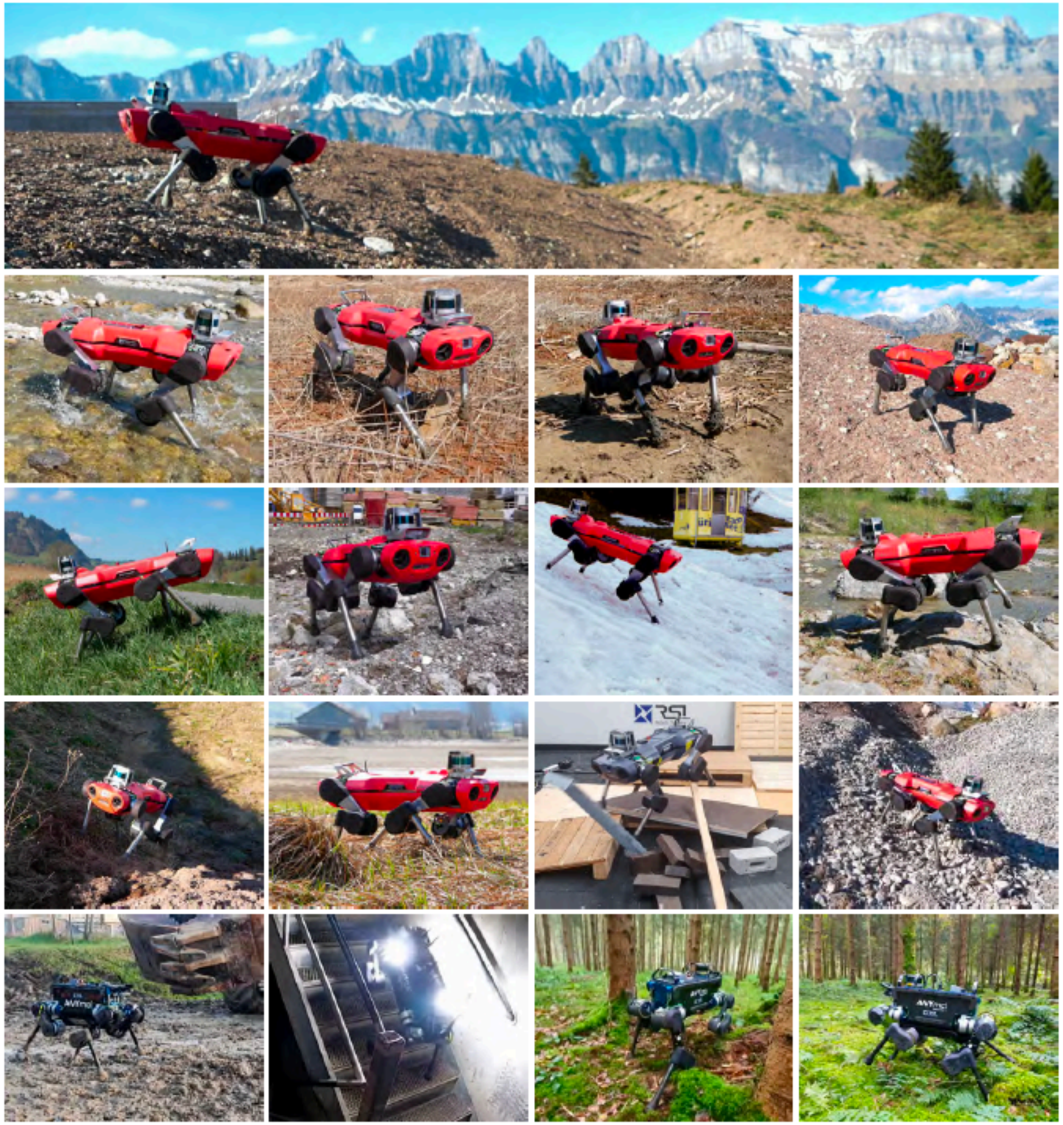


Fig. 1. Deployment of the presented locomotion controller in a variety of challenging environments.

Learning Quadrupedal Locomotion over Challenging Terrain, Joonho Lee et. al., 2020

Real-World Robotic System Design: Quadrupedal

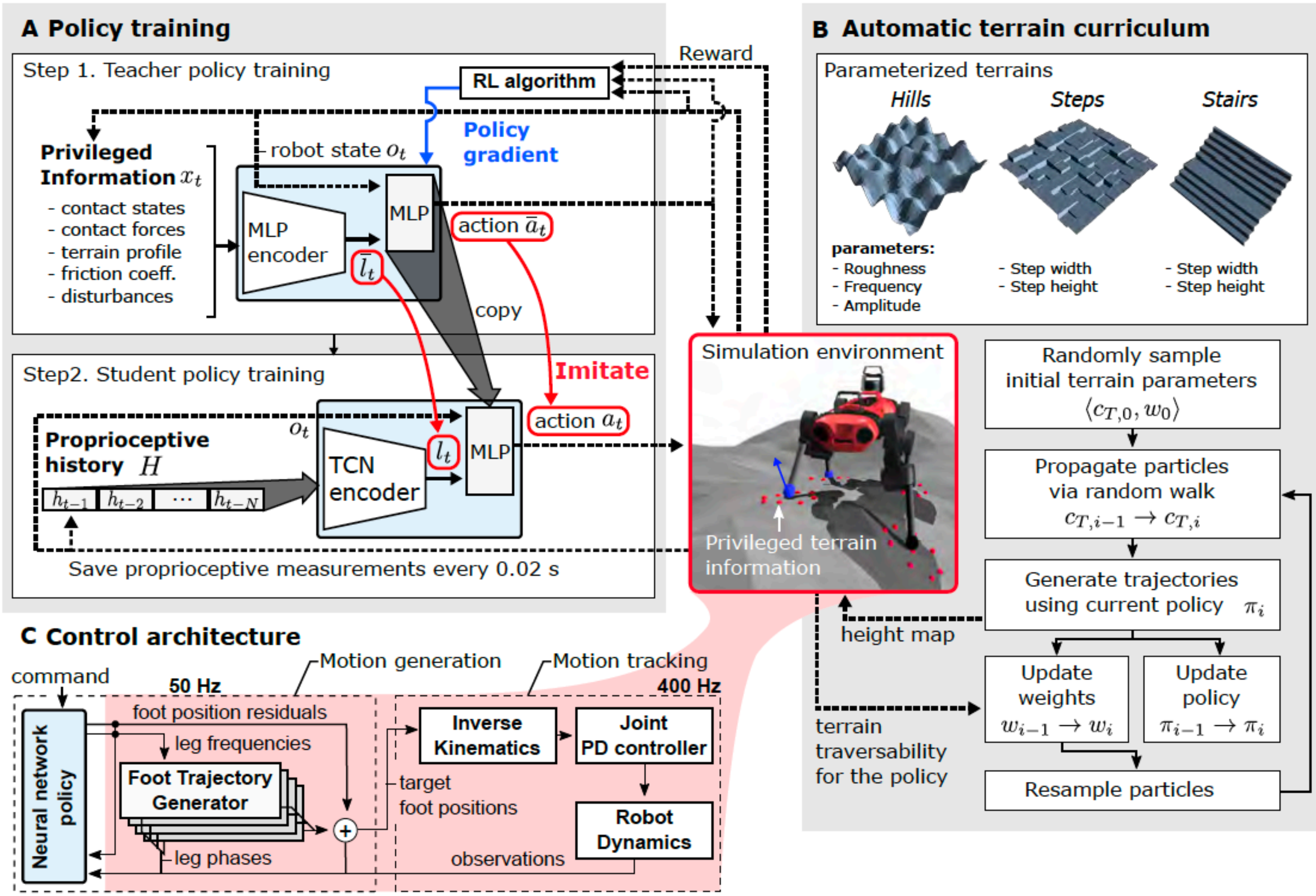
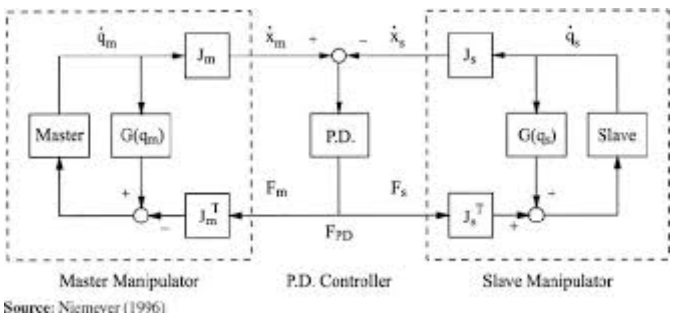


Fig. 4. Overview of the presented approach. (A) Two-stage training process. First, a teacher policy is trained using reinforcement learning in simulation. It has access to privileged information that is not available in the real world. Next, a proprioceptive student policy learns by imitating the teacher. The student policy acts on a stream of proprioceptive sensory input and does not use privileged information. (B) An adaptive terrain curriculum synthesizes terrains at an appropriate level of difficulty during the course of training. Particle filtering is used to maintain a distribution of terrain parameters that are challenging but traversable by the policy. (C) Architecture of the locomotion controller. The learned proprioceptive policy modulates motion primitives via kinematic residuals. An empirical model of the joint PD controller facilitates deployment on physical machines.

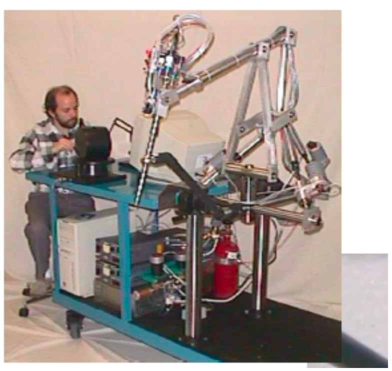
Learning from Imitation

OC/DP

Adaptive control
Inverse Lyapunov
Leader/follower

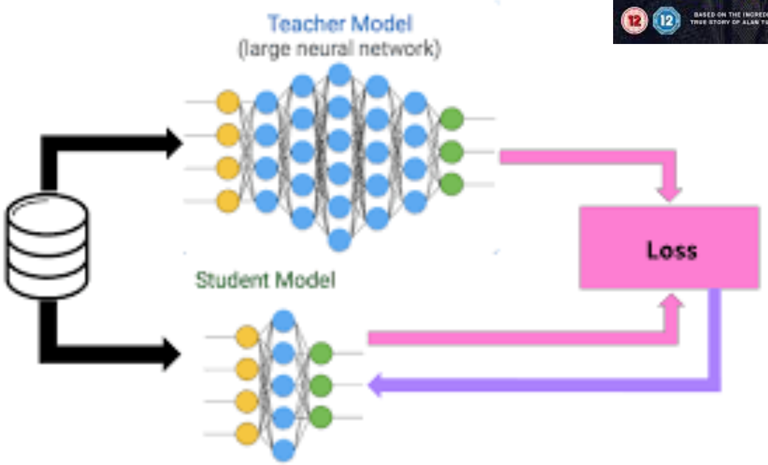


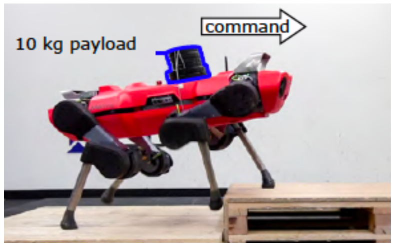
Telesurgery Workstation 1999
Sastry



AI/RL

Imitation learning
Inverse RL
Teacher/student



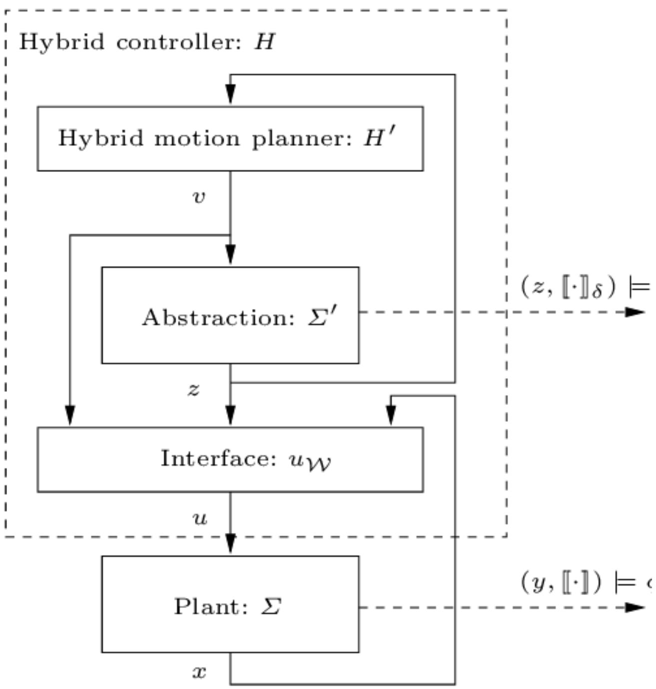


Learning Quadrupedal Locomotion over Challenging Terrain, Joonho Lee et. al., 2020

Hierarchical Design and Control Architecture

OC/DP

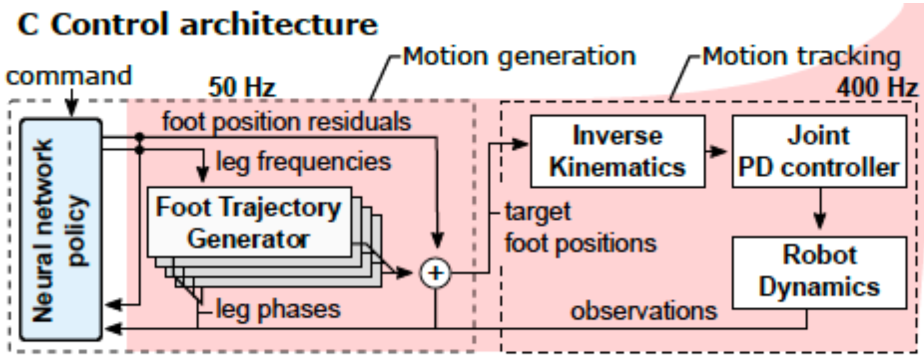
Hierarchical synthesis
Hybrid controllers



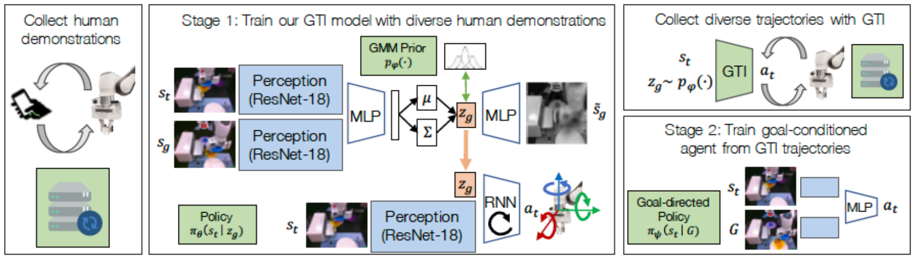
Hierarchical Synthesis of Hybrid Controllers from Temporal Logic Specifications, Georgios Fainekos et. al. 2007

AI/RL

Generalized imitation



Learning Quadrupedal Locomotion over Challenging Terrain, Joonho Lee et. al., 2020



Learning to Generalize Across Long-Horizon Tasks from Human Demonstrations, Ajay Mandlekar et. al. 2020…

Key Challenges and Guidelines

Bridge Principles and Practices via Computation:

1. **Scalabiltiy meets Low-dimensionality:**
   - Q-Learning versus Matrix Completion
   - A "Compressive Sensing Theory" for learning to control?

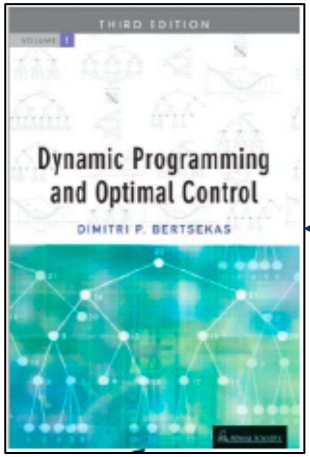
2. **System/domain Adaptation/transfer:**
   - Adaptive Control versus Imitation Learning.
   - Leader/follower versus Teacher/student

3. **Complexity meets Hierarchical Abstraction:**
   - Hierarchical and hybrid control design versus
   - High-level/low-level or long-term/short-term learning

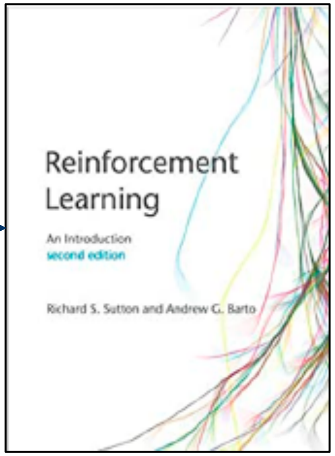
4. **Quantitative objectives versus Qualitative goals**
   - Time, energy, cost, precision (OC/DP)
   - Stability, survivability, or winning (Control/RL)
   - A "Lyapunov Theory" for learning to achieve qualitative goals?

Questions, please?

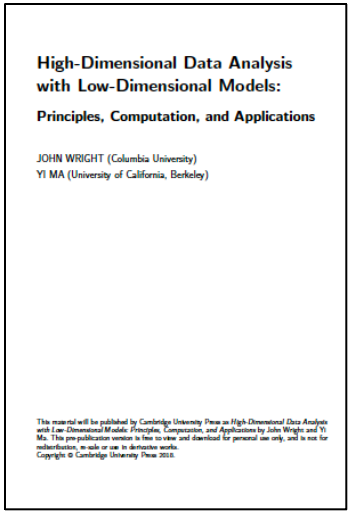
Principles



Practices



Computation



High-Dimensional Data Analysis with Low-Dimensional Models:

Principles, Computation, and Applications

JOHN WRIGHT (Columbia University)
YI MA (University of California, Berkeley)

This material will be published by Cambridge University Press as *High-Dimensional Data Analysis with Low-Dimensional Models: Principles, Computation, and Applications* by John Wright and Yi Ma. This pre-publication version is free to view and download for personal use only, and is not for redistribution, re-sale or use in derivative works.
Copyright © Cambridge University Press 2018.

Practice Keeps Theory Honest, and Vice Versa!