Robotics
Spring 2023

Abhishek Gupta

TAs: Yi Li, Srivatsa GS

Recap: Course Overview

Filtering/Smoothing
Localization
Mapping
SLAM

Search
Motion Planning
TrajOpt
Stability/Certification

MDPs and RL
Imitation Learning
Solving POMDPs

Lecture Outline

Reinforcement Learning: Motivation

↓

Imitation Learning

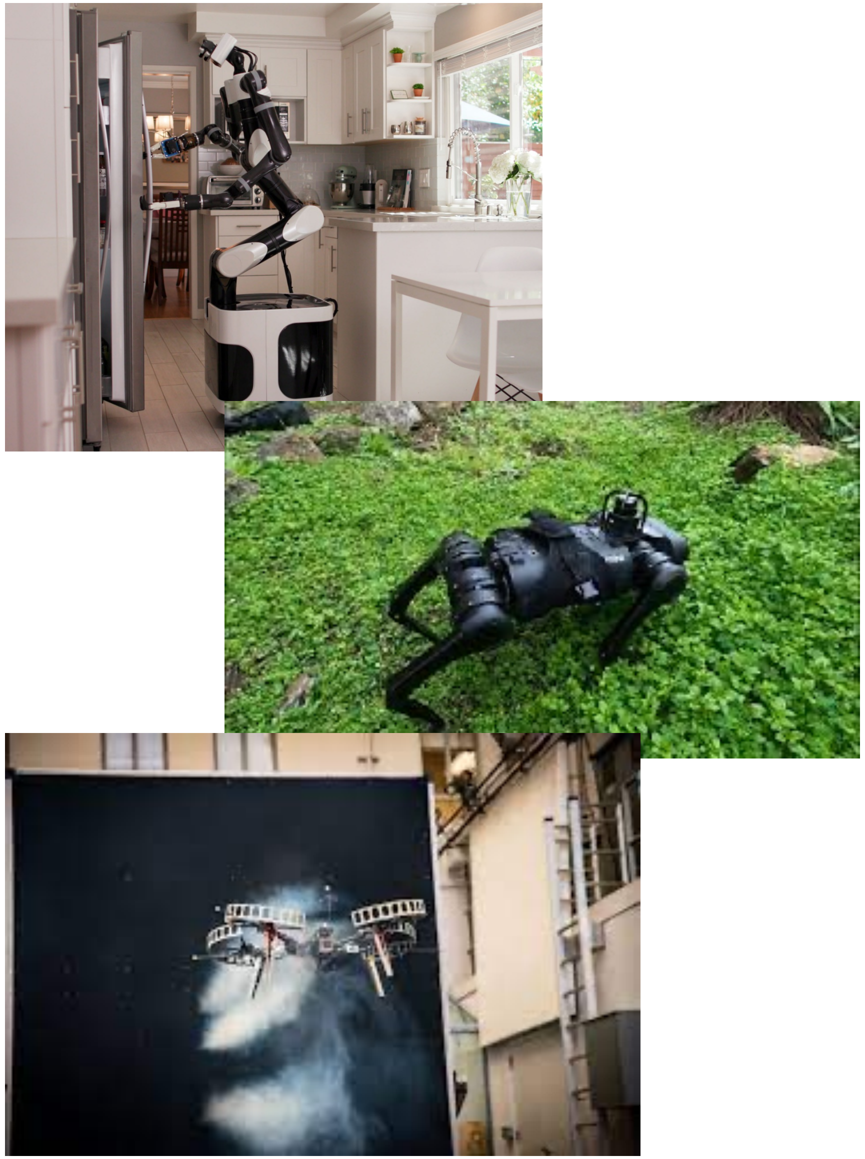
↓

Policy Gradient and Beyond

When is optimal control + planning not enough?

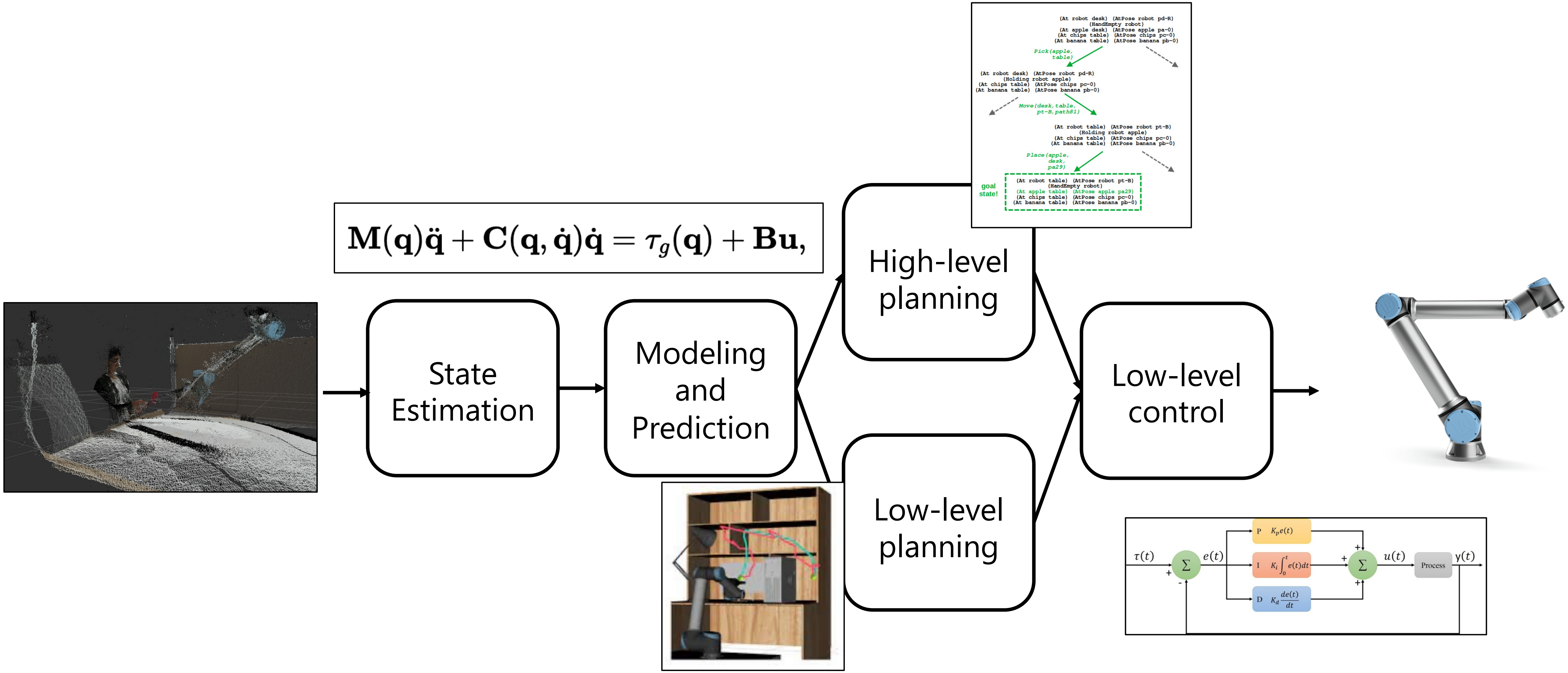
Unstructured Environments

Diversity — Variability — Complexity

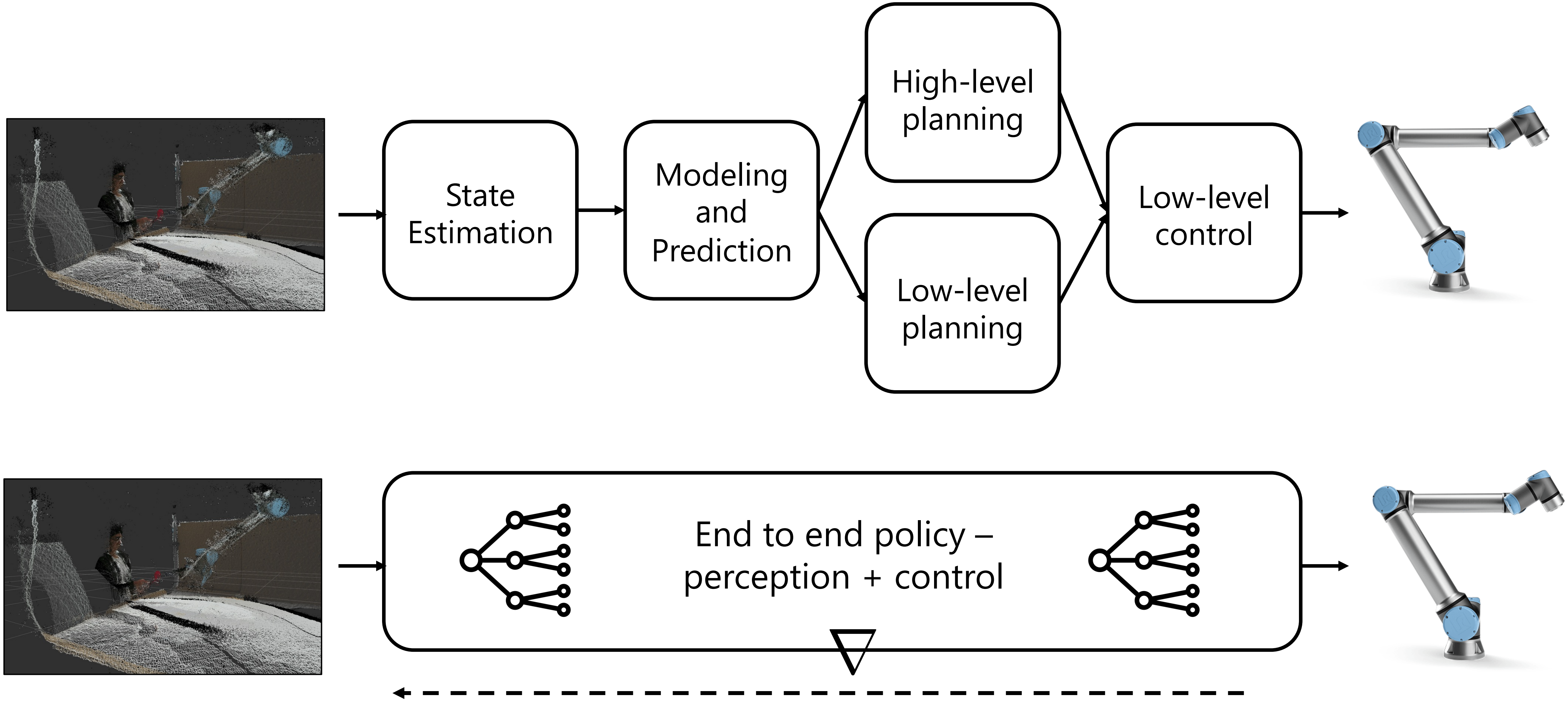


Challenging to model these environments

How does a typical robotics pipeline look?

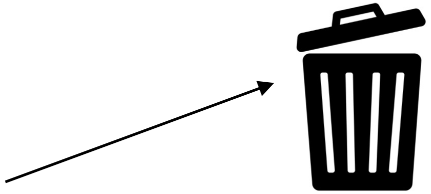


Deep reinforcement learning pipeline for robotics

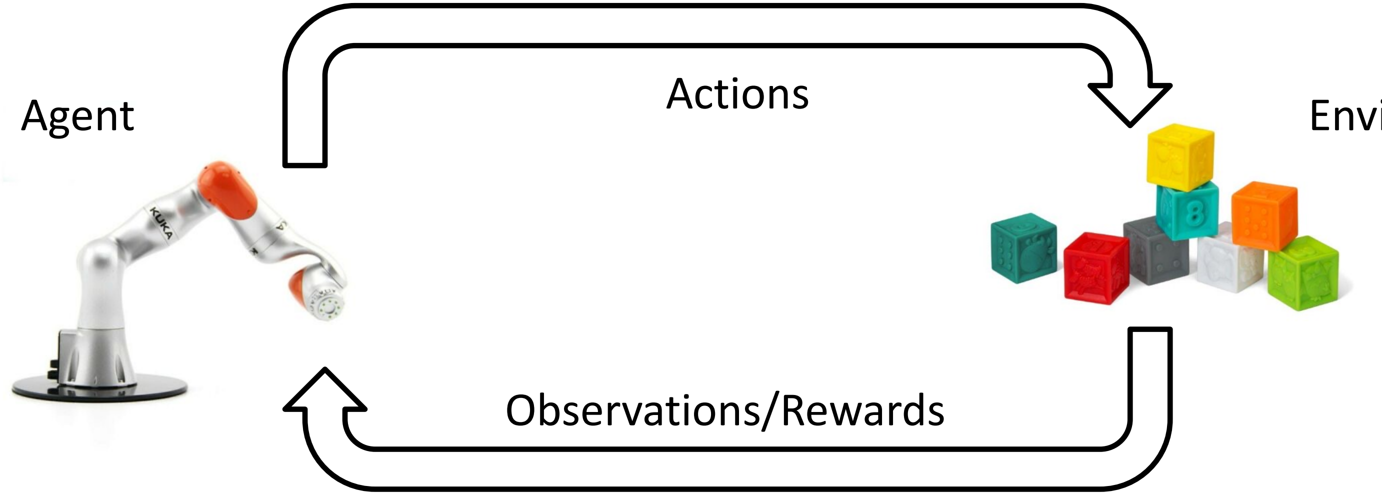


What is reinforcement learning?

Remove assumption for known environment model, learn directly from data



Modeling assumptions



Agent has:

- Sensing
- Actuation

Environment accepts:

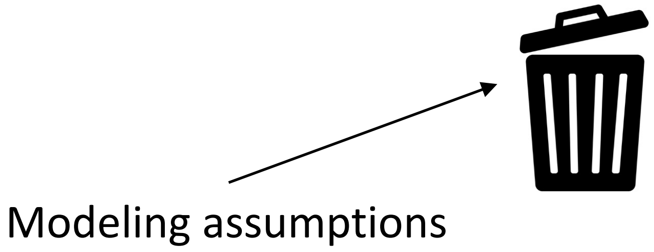
- Actions from agent

Produces

- Observations for sensors(usually unknown)

Why would you do this?

Remove assumption for known environment model, learn directly from data



Pros:

1. Continual improvement on deployment
2. Avoid significant modeling assumptions and simulation
3. Scale across tasks easily!

Cons:

1. Potentially prohibitive data requirements
2. Sometimes unstable, lacks guarantees
3. Poor extrapolation

Promising and useful tool in unstructured, dynamic environments

Connection to Optimal Control

Closely related: typically problem of finding control given a plant model

$$\min_{x,u} \int_0^X L(t, x(t), u(t)).dx$$

w.r.t

$$x'(t) = f(x(t), u(t))$$

$$\max_\theta \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} r(s_t, a_t) \right]$$

Main difference: model known vs unknown
Minor differences: Cost vs reward, discrete vs continuous time

Is it useful for robotics?

Robots that get better over time, adapting to new **environments**

iRobot

Is it useful for robotics?

Robots that get better over time, adapting to new ***objects***

Kalashnikov et al 18

Is it useful for robotics?

Robots that get better over time, adapting to new **terrains**

Miki et al 22

Is it useful for robotics?

Robots that get better over time, adapting to new **tasks**

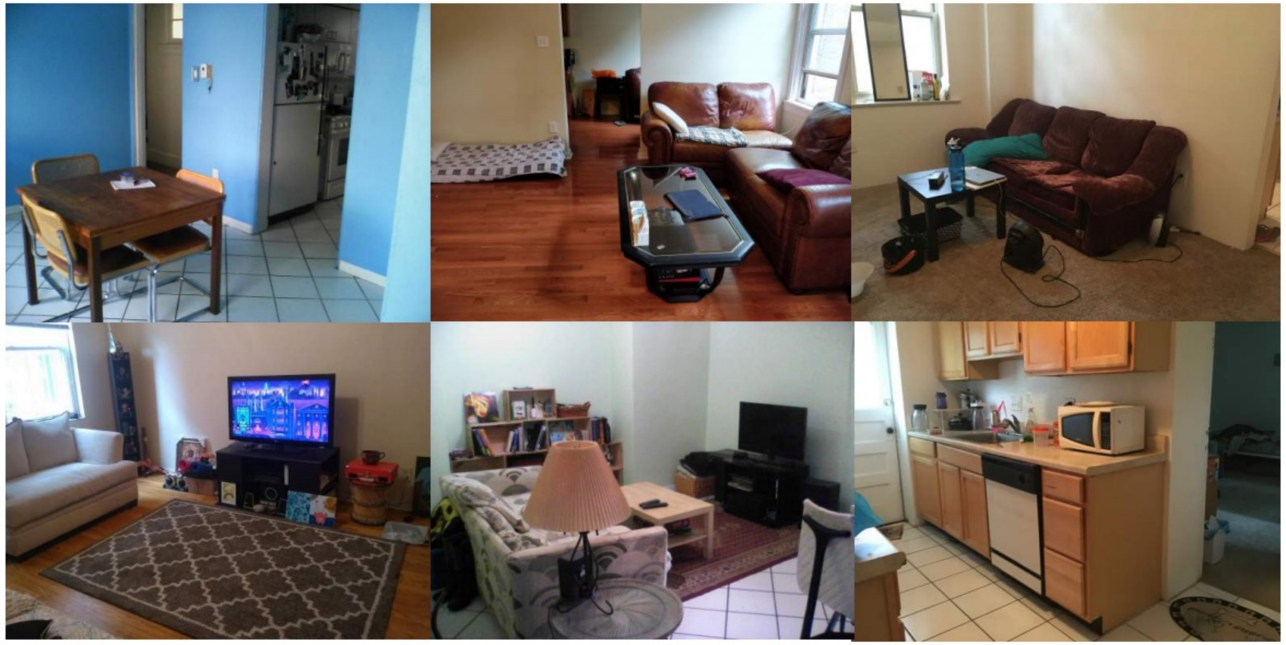
Gupta et al 21

Why should we care about RL?

Allows agents to continue improving/adapting on deployment with minimal human effort

Locomotion

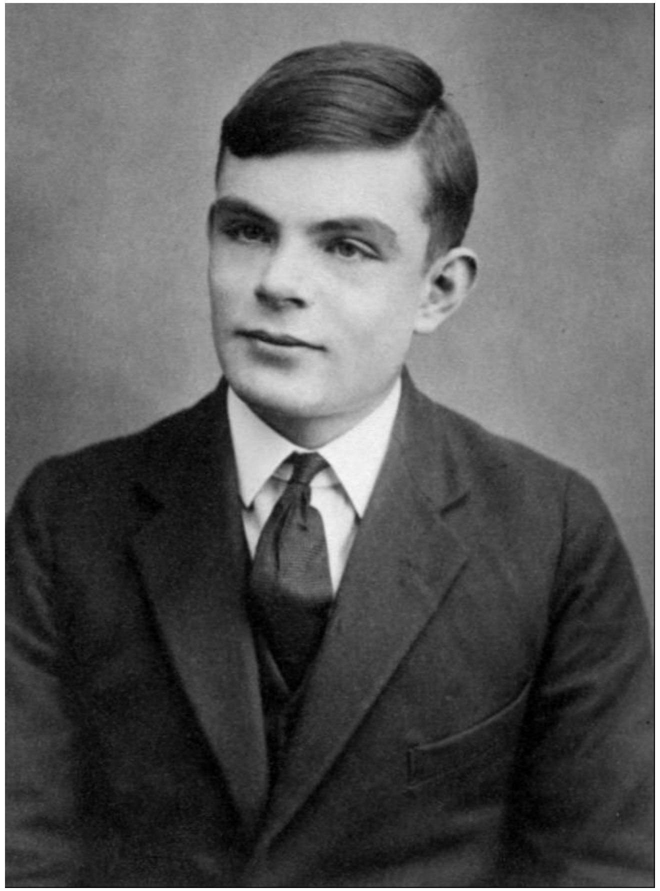
Manipulation



Agents can overfit to domains at test time rather than per-domain human design

Why should we care about RL?

Hypothesis: By designing algorithms that can improve themselves, we can reach fully intelligent systems

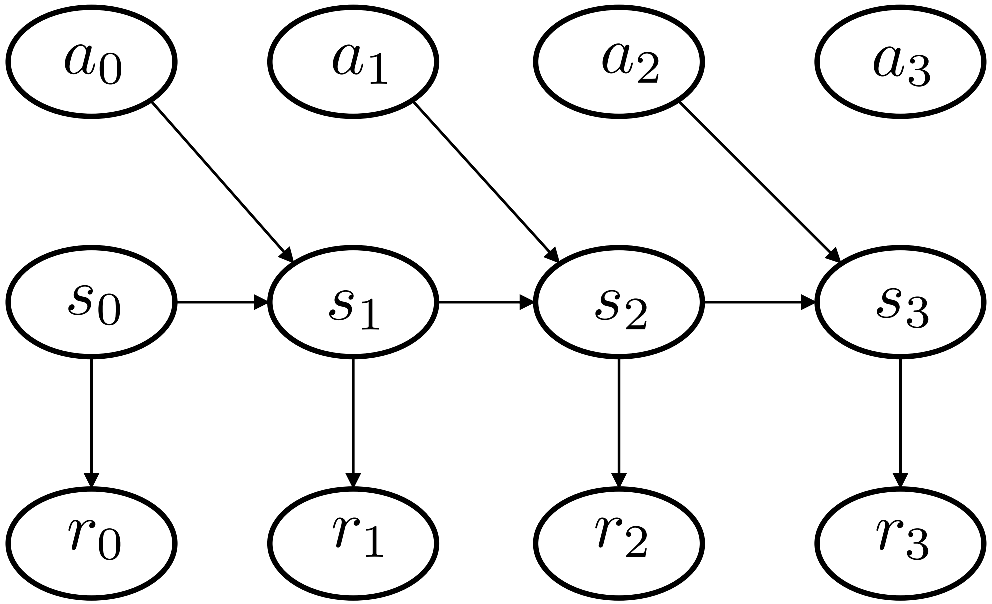


"Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's? If this were then subjected to an appropriate course of education one would obtain the adult brain" – Alan Turing

⇓

Rather than try to directly replicate behaviors, try to replicate adaptative learning mechanisms

Markov Decision Processes



States: $\mathcal{S}$

Actions: $\mathcal{A}$

Rewards: $\mathcal{R}$

Transition Dynamics - $p(s_{t+1}|s_t, a_t)$

Markov property $\quad p(s_1, s_2, s_3) = p(s_3|s_2)p(s_2|s_1)p(s_1)$

Trajectory $\quad \tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_T, a_T, r_T)$

MDPs to the Real World

Task: Place kettle in sink

State: Camera Images / Joint Encoders

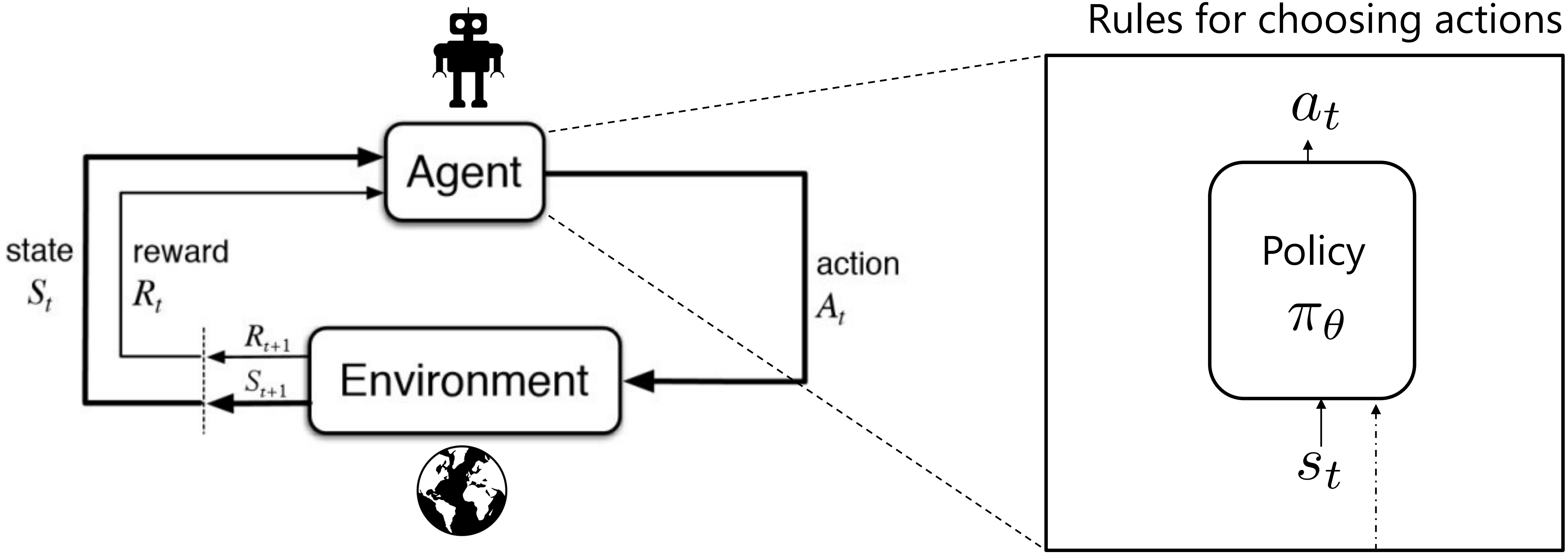
Action: Joint torques/velocities

Reward: Distance from kettle to sink

Transition: World physics

Reinforcement Learning Formalism

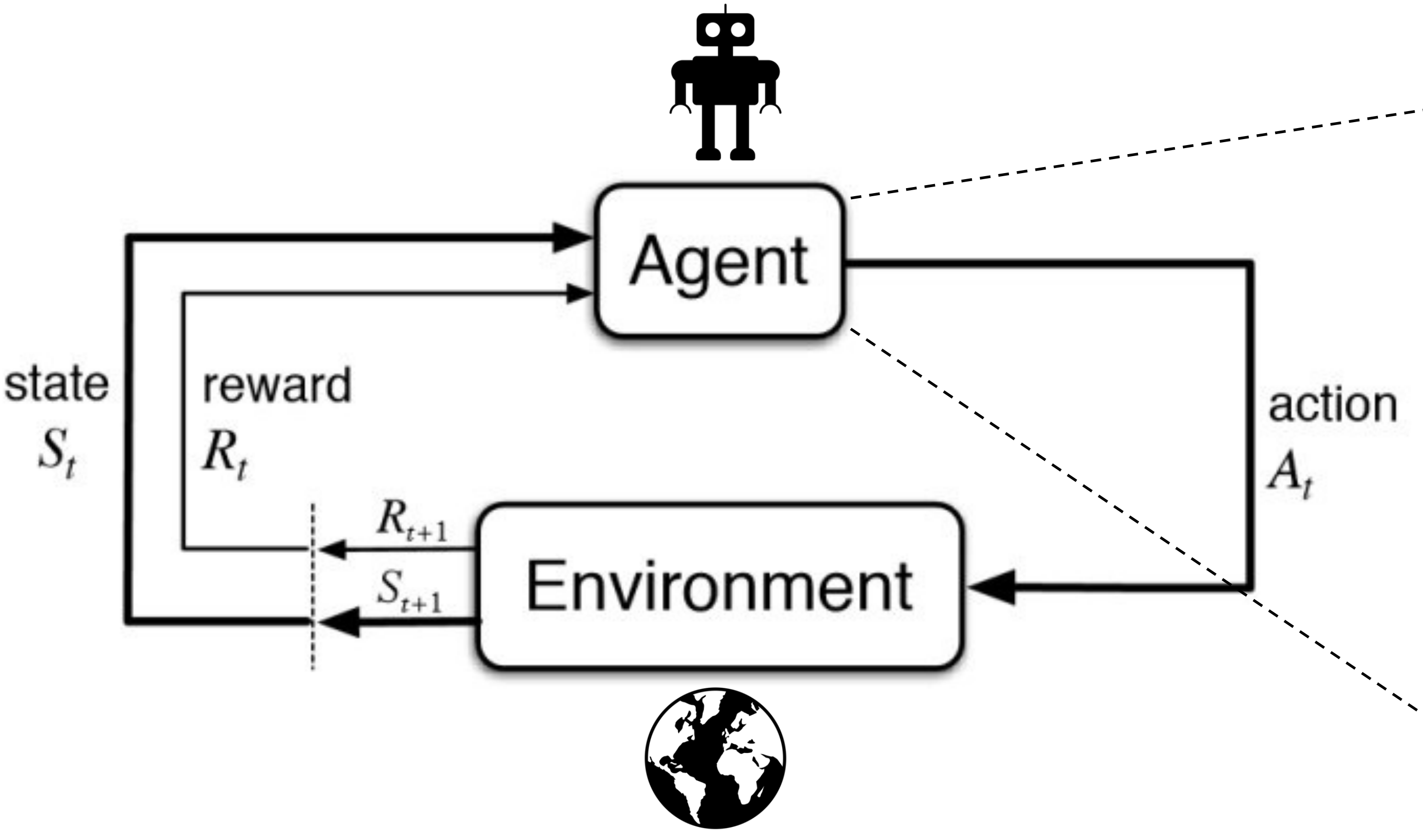
Rules for choosing actions



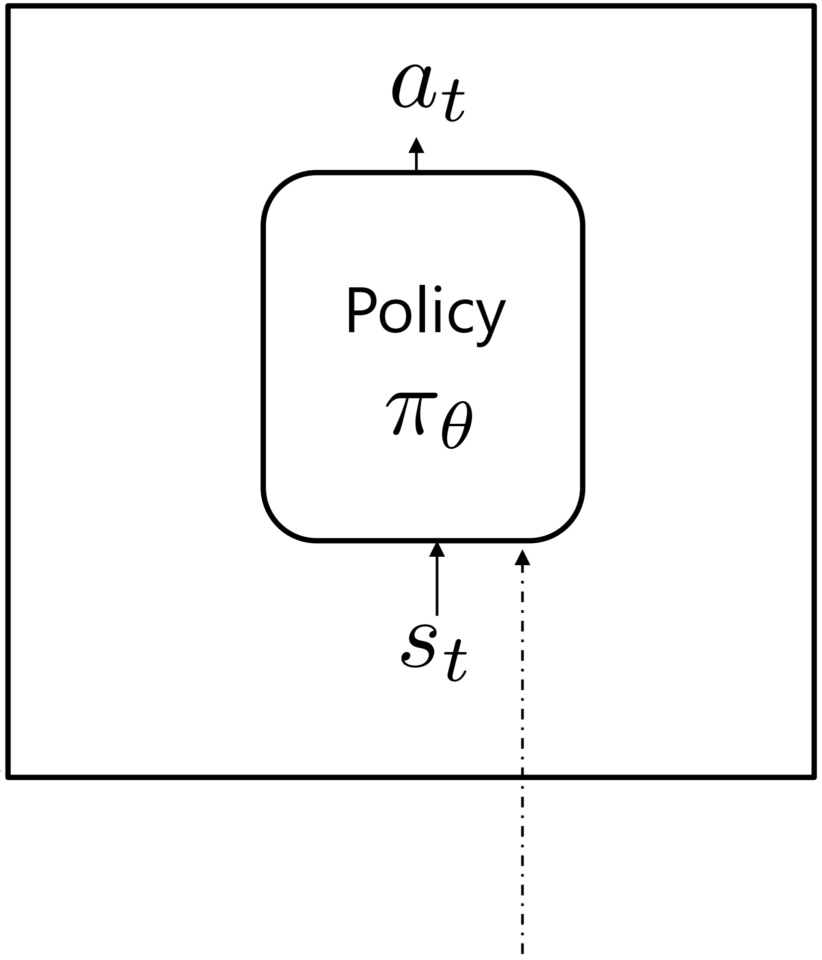
Maximize the sum of expected rewards under policy

Needs to be learned

Reinforcement Learning Formalism



Rules for choosing actions



$$\max_\theta \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} r(s_t, a_t) \right]$$

Trajectory sampled using policy

Needs to be learned

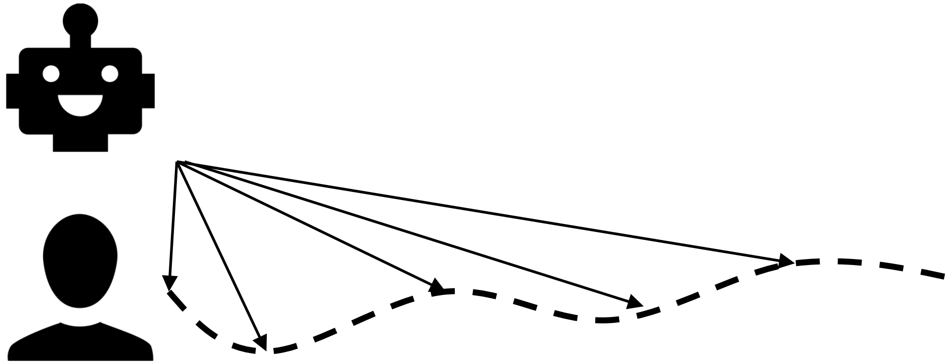
Why is this not just supervised learning?

Supervised Learning

$$\max_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\log \hat{p}_\theta(y|x)\right]$$

Sampling from expert

$D_{\mathrm{KL}}(p^*||p_\theta)$ IID

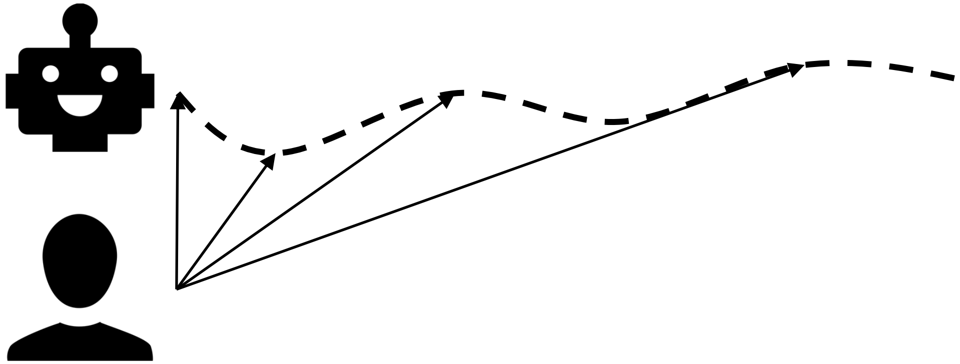


Reinforcement Learning

$$\max_\theta \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} r(s_t, a_t)\right]$$

Sampling from policy

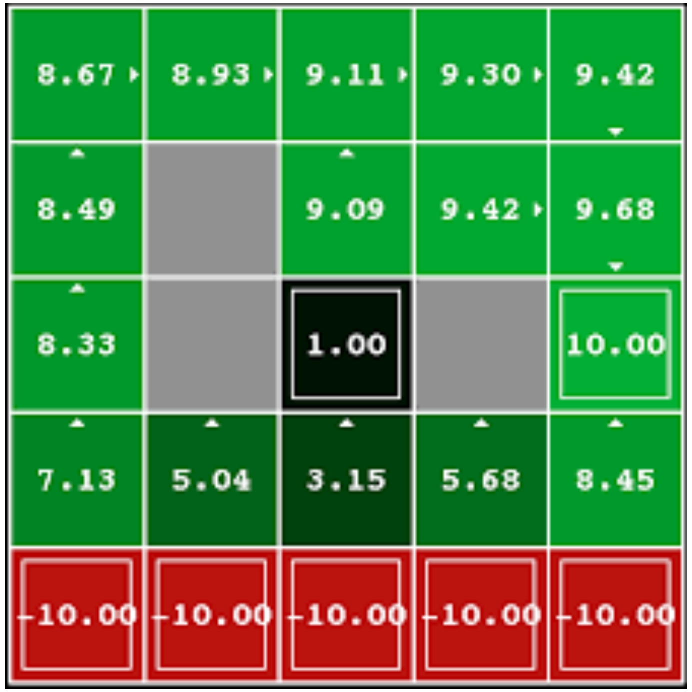
$D_{\mathrm{KL}}(p_\theta||p^*)$ Non-IID



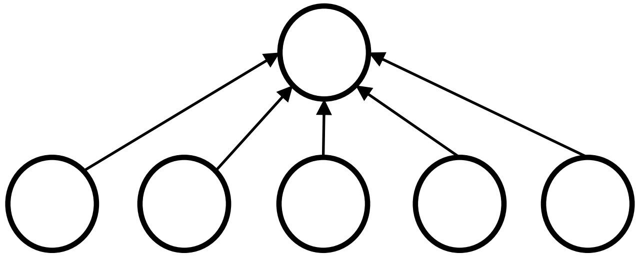
Main thing to learn - Policies

Policies are mappings from states to optimal actions

Tabular

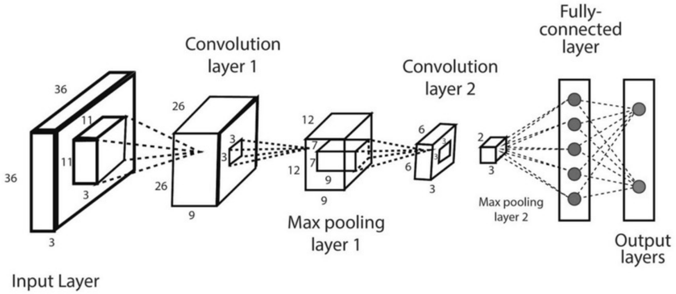


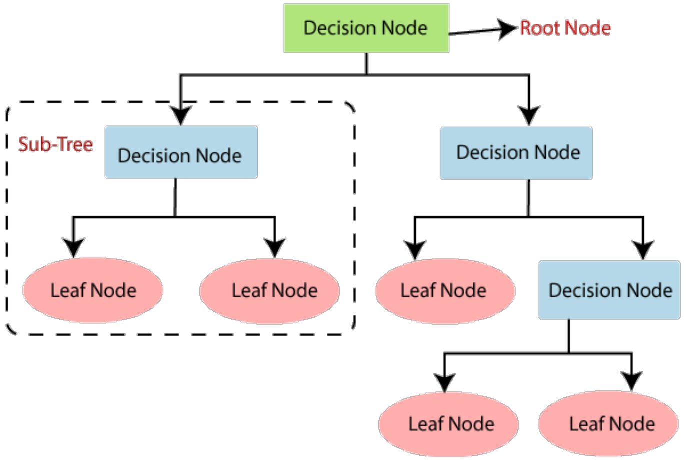
Linear



$$\pi(a|s) = \langle \phi(s, a), w \rangle$$

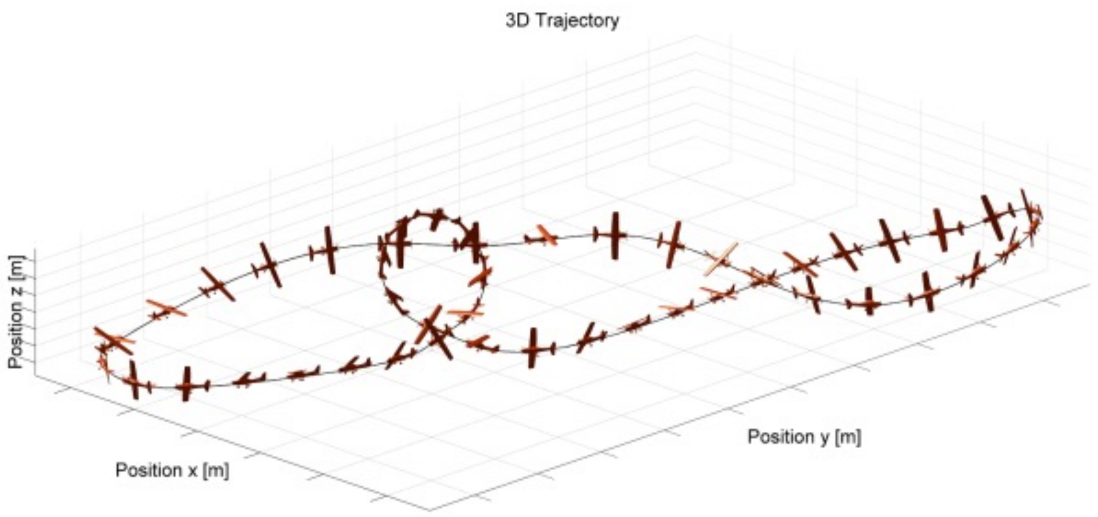
Arbitrary function approx





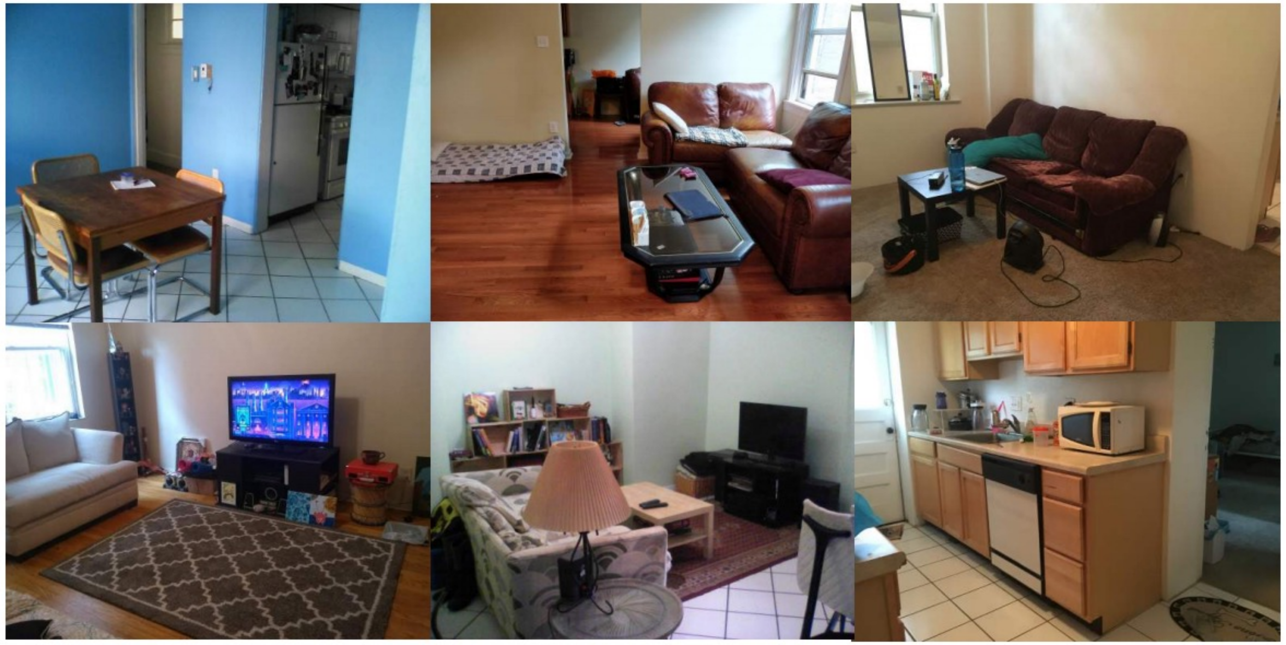
Where is Reinforcement Learning not useful?

Not the right call for very safety-critical, repetitive applications



Where is Reinforcement Learning "potentially" useful?

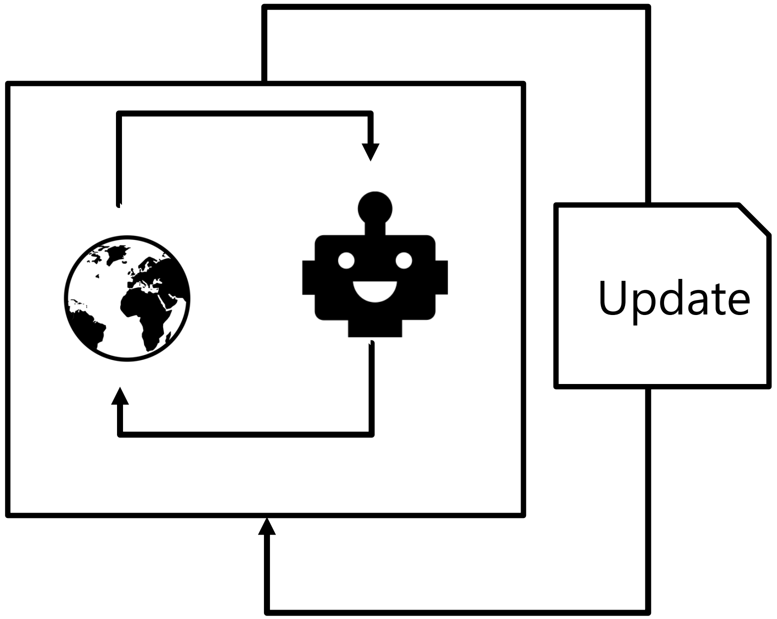
Domains which have high diversity, yet relatively cheap autonomous data collection



But these domains are not as simple as just running RL algorithms!

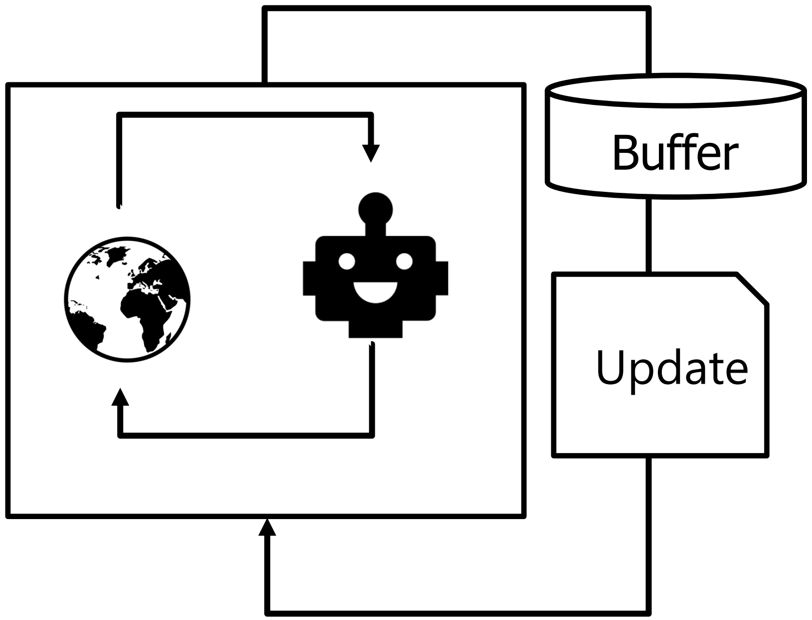
Learning Algorithms for Robotics

On-Policy Algorithms



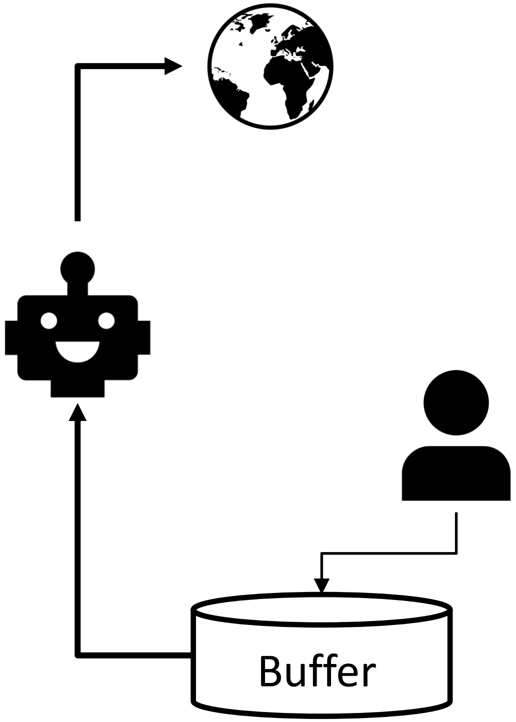
Simple, performant,
Data inefficient

Off-Policy Algorithms



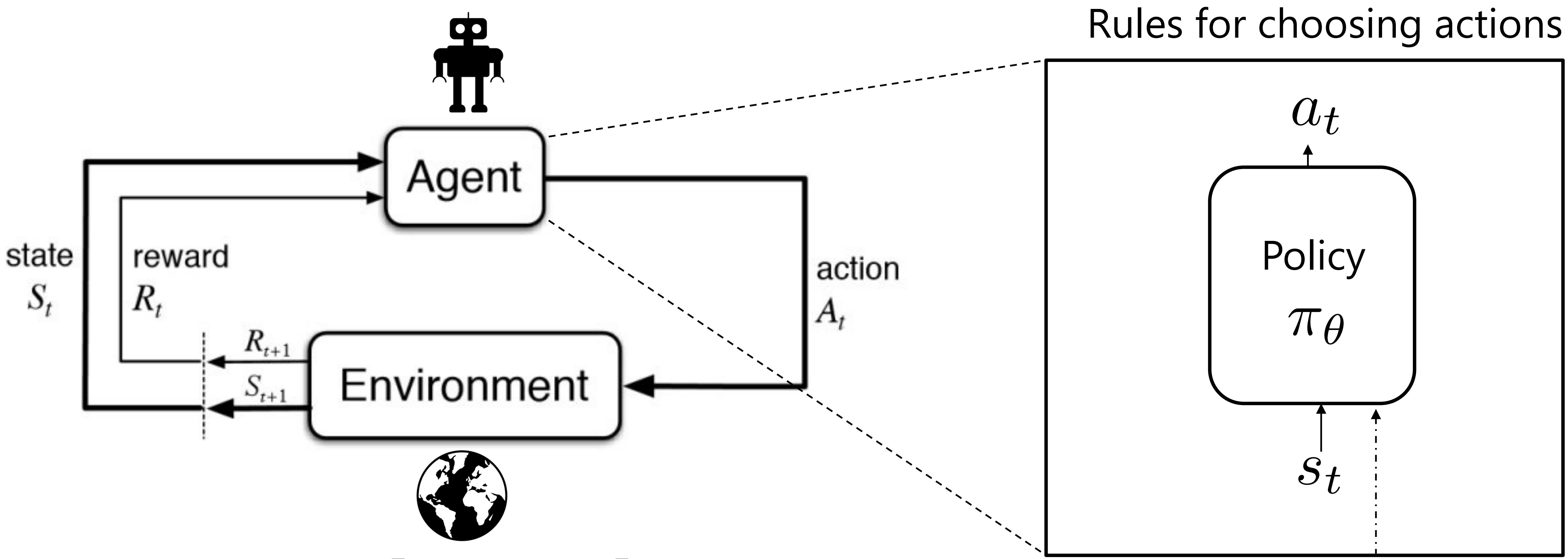
Data-efficient,
sometimes unstable

Imitation Learning



Performant, efficient, but
compounding error and
expensive data collection

Objective of Reinforcement Learning

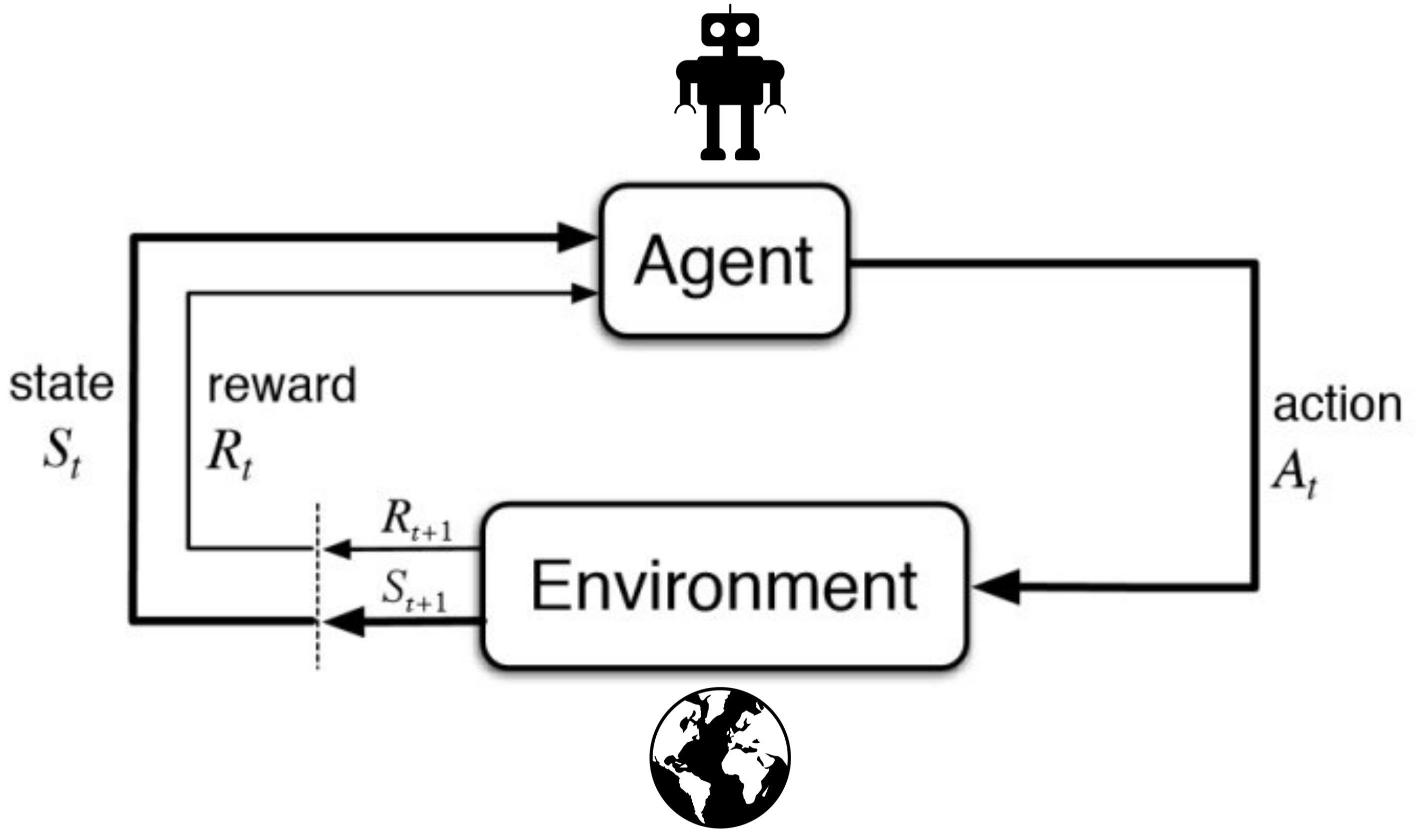


$$\max_\theta \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} r(s_t, a_t) \right]$$

Trajectory sampled using policy

Needs to be learned

Objective of Reinforcement Learning



$$\max_{\theta} \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} r(s_t, a_t) \right]$$

Trajectory sampled using policy

Assumptions:

1. Rewards are additive
2. Dynamics can be sampled from, but functional form is unknown
3. Rewards are provided as every state is visited, functional form is unknown

Lecture Outline

Reinforcement Learning: Motivation

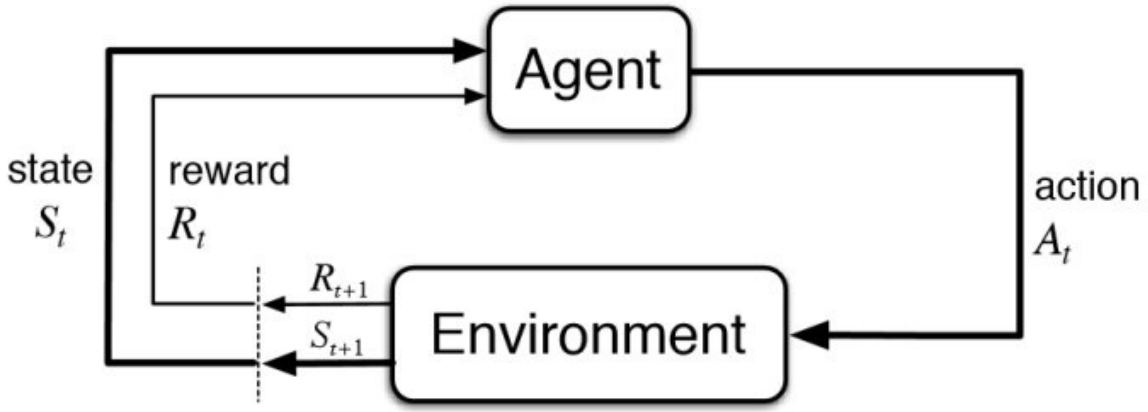
↓

Imitation Learning

↓

Policy Gradient and Beyond

What makes RL hard?



Exploration: Find the right data

Exploitation: Using the exploration data

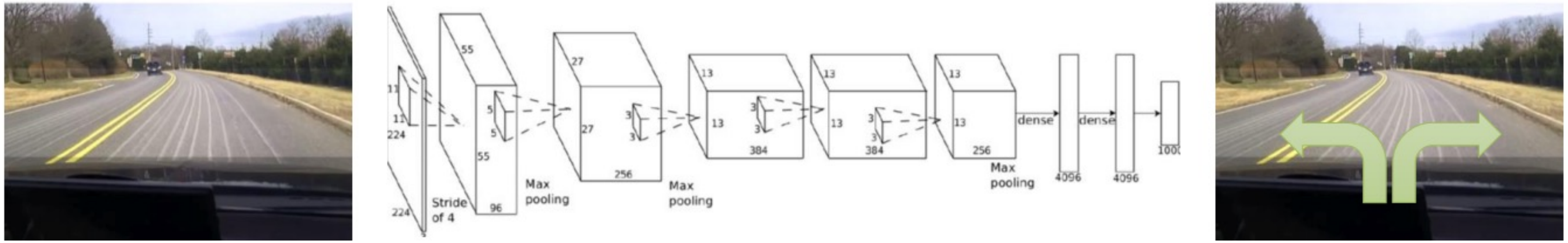
Exploration is challenging in itself in large state spaces, and exploration + exploitation is particularly tough

→ let's avoid for now!

Imitation Learning (IL) in a Nutshell

Given Demonstration

Goal a policy to mimic the demonstrator



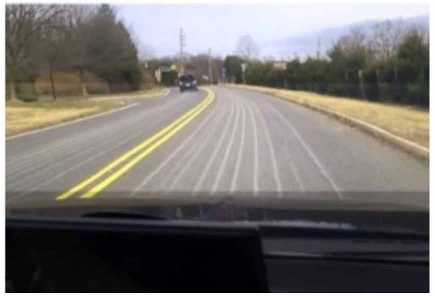
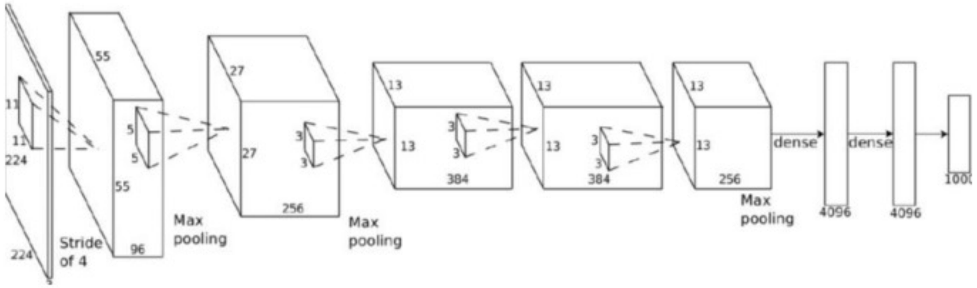
Credit: Deep Reinforcemet Learning, Sergey Levine.

Behavior Cloning

Given demonstration $D^* = s_1^*, a_1^*, s_2^*, a_2^*, \ldots$

Optimize $\arg\min \mathrm{E}_{(s_i^*, a_i^*) \sim D^*} L(a_i^* \mid \pi(a|s_i^*))$

e.g. $(\pi(s_i^*) - a_i^*)^2$

s_t $\pi_\theta(a_t|s_t)$ a_t

Credit: Deep Reinforcement Learning, Sergey Levine.

ALVINN: Autonomous Land Vehicle In a Neural Network

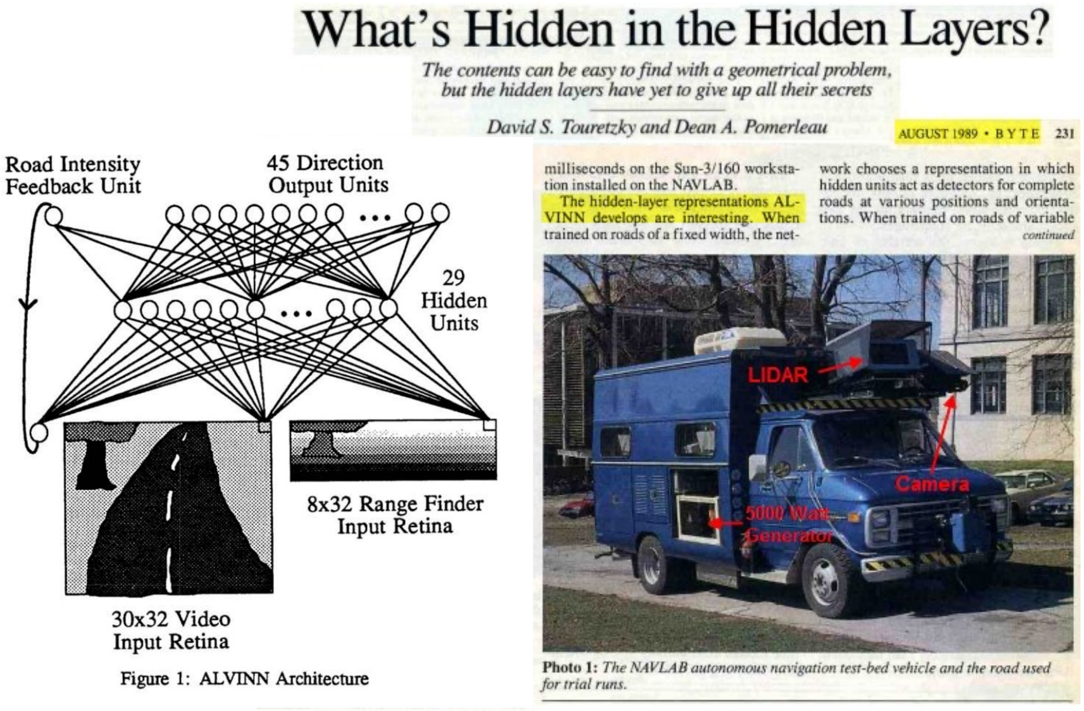


Figure 1: ALVINN Architecture

What's Hidden in the Hidden Layers?

The contents can be easy to find with a geometrical problem, but the hidden layers have yet to give up all their secrets

David S. Touretzky and Dean A. Pomerleau

AUGUST 1989 • BYTE 231

milliseconds on the Sun-3/160 workstation installed on the NAVLAB.

The hidden-layer representations ALVINN develops are interesting. When trained on roads of a fixed width, the network chooses a representation in which hidden units act as detectors for complete roads at various positions and orientations. When trained on roads of variable *continued*

Photo 1: *The NAVLAB autonomous navigation test-bed vehicle and the road used for trial runs.*

Covariate Shift

- Imitation Learning ≠ Supervised Learning

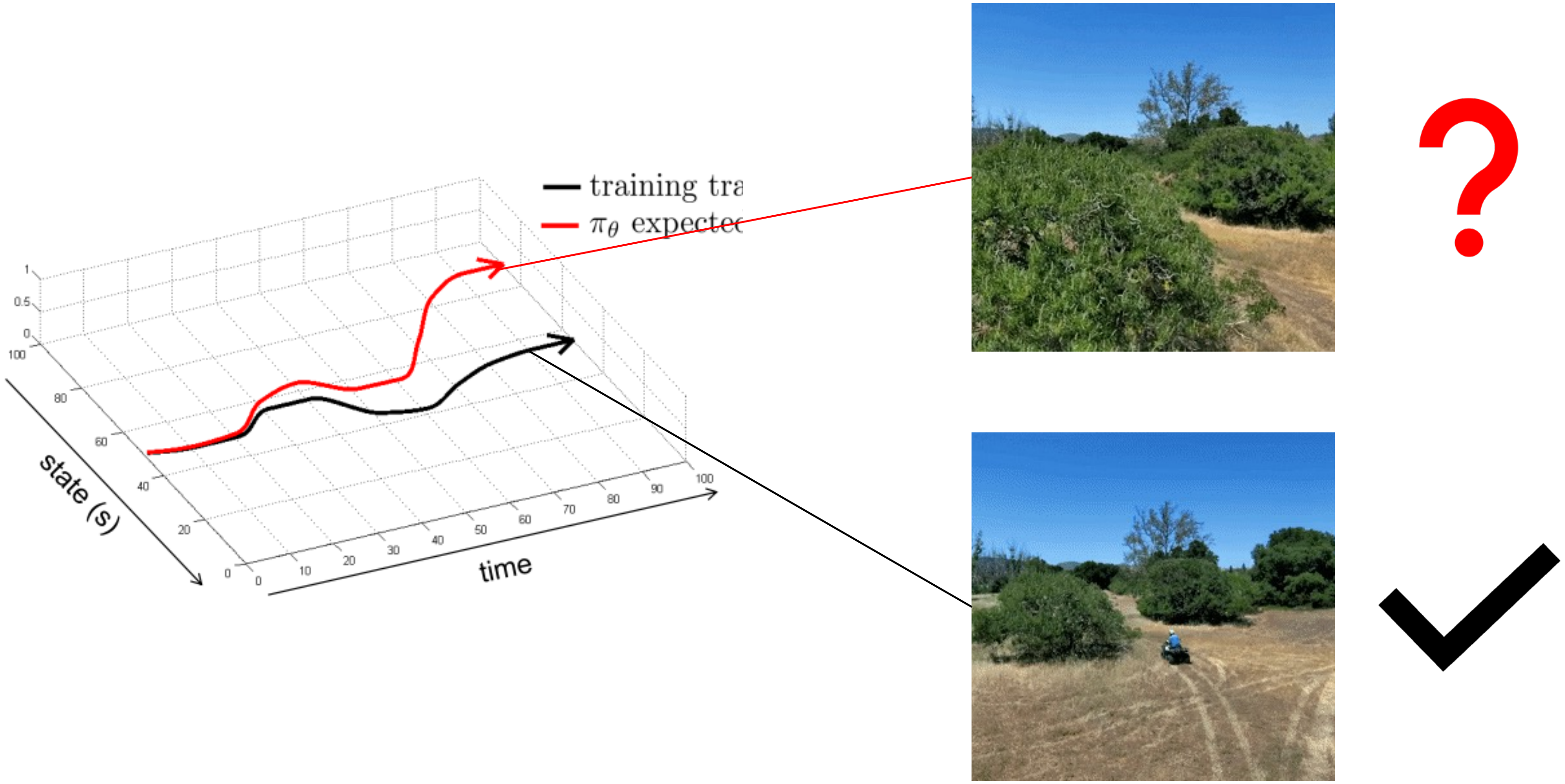
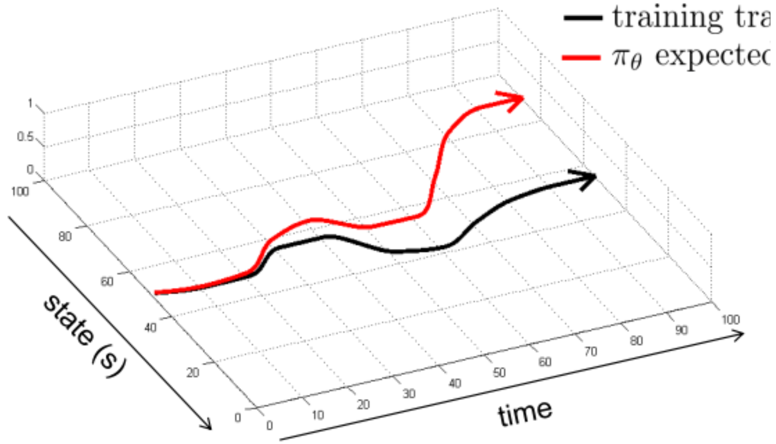


Image Credit: Deep Reinforcement Learning. Sergey Levin.
Video Credit: Skydio Autonomy, “Deep Neural Pilot on Skydio 2”

Covariate Shift

- Imitation Learning ≠ Supervised Learning



$$\arg\min \mathrm{E}_{(s_i^*, a_i^*) \sim D^*} L(a_i^* \mid \pi(a|s_i^*))$$

Image Credit: Deep Reinforcement Learning. Sergey Levin.
Video Credit: Skydio Autonomy, “Deep Neural Pilot on Skydio 2”

Learn to Recover from Mistakes

Is there a more general purpose solution?

DAgger

Reformulate Imitation Learning as an Online Learning problem

$$p_\theta(s_t) \rightarrow p_{\text{train}}(s_t)$$

DAgger: Dataset Aggregation, 2011

Initialize $\hat{\pi}_1$ to any policy in Π.
for $i = 1$ to N do
    Sample trajectories using $\hat{\pi}_i$.
    Query expert for labels: $\mathcal{D}_i = \{(s, \pi^*(s))\}$ of visited states
    Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{D}_i$.
    Train classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$.

Query Expert

Combats covariate shift by bringing p_θ and p_{train} together

Credit: "A Reduction of Imitation Learning and Structure Prediction to No-Regret Online Learning" Ross 2011. Video Credit: The Bird MURI project. Drew Bagnell's group.

Lecture Outline

Reinforcement Learning: Motivation

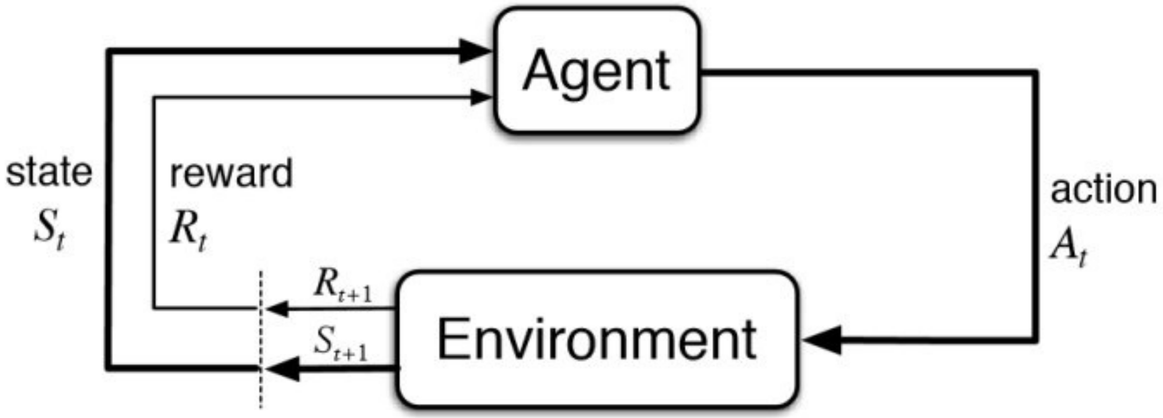
↓

Imitation Learning

↓

Policy Gradient and Beyond

Let's revisit the overall RL problem



$$\max_\theta \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} r(s_t, a_t) \right]$$

Gradient Ascent

Dynamic Programming

Model-Based Optimization

Each method has it's own +/-

What if we just performed gradient ascent?

$$\max_\theta \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} r(s_t, a_t) \right]$$

$$= \int p_\theta(\tau) R(\tau) d\tau$$

Standard gradient descent (supervised learning)

$$\nabla_\theta \mathbb{E}_{x \sim g(x)} \left[f_\theta(x) \right]$$

REINFORCE gradient descent (RL)

$$\nabla_\theta \mathbb{E}_{x \sim p_\theta(x)} \left[f(x) \right]$$

Gradient wrt expectation variable, not of integrand!

Taking the gradient of sum of rewards

$$J(\theta) = \int p_\theta(\tau) R(\tau) d(\tau)$$

$$\nabla_\theta J(\theta) = \nabla_\theta \int p_\theta(\tau) R(\tau) d(\tau)$$

$$= \int \nabla_\theta p_\theta(\tau) R(\tau) d(\tau) \quad = \int \frac{p_\theta(\tau)}{p_\theta(\tau)} \nabla_\theta p_\theta(\tau) R(\tau) d(\tau)$$

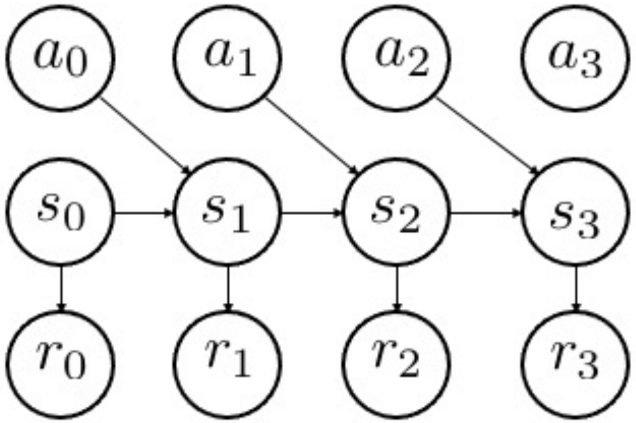
$$= \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) R(\tau) d(\tau) \ = \mathbb{E}_{p_\theta(\tau)} \left[\nabla_\theta \log p_\theta(\tau) R(\tau) \right]$$

REINFORCE trick

Taking the gradient of return

Initial State — Dynamics — Policy

$$p_\theta(\tau) = p(s_0)\Pi_{t=0}^{T-1} p(s_{t+1}|s_t, a_t)\pi(a_t|s_t)$$



$$\log p_\theta(\tau) = \log p(s_0) + \sum_{t=0}^{T-1} \log p(s_{t+1}|s_t, a_t) + \log \pi(a_t|s_t)$$

$$\nabla_\theta \log p_\theta(\tau) = \nabla_\theta \log p(s_0) + \sum_{t=0}^{T-1} \nabla_\theta \log p(s_{t+1}|s_t, a_t) + \nabla_\theta \log \pi(a_t|s_t)$$

$$\nabla_\theta \log p_\theta(\tau) = \sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t)$$

Model Free!!

Taking the gradient of return

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\nabla_\theta \log p_\theta(\tau) \sum_{t=0}^{T} r(s_t, a_t) \right]$$

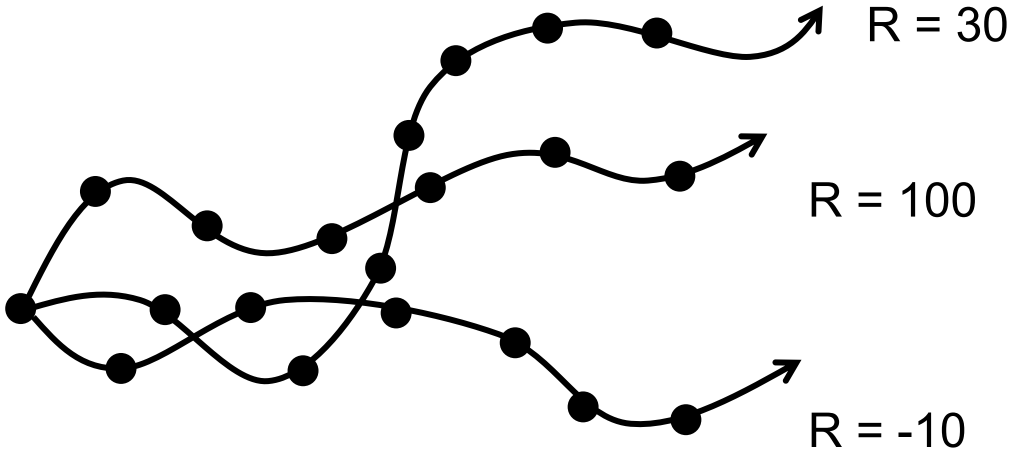
$$\nabla_\theta J(\theta) = \mathbb{E}_{\substack{s_0 \sim p(s_0) \\ s_{t+1} \sim p(s_{t+1}|s_t, a_t) \\ a_t \sim \pi(a_t|s_t)}} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t'=0}^{T} r(s_t, a_t) \right]$$

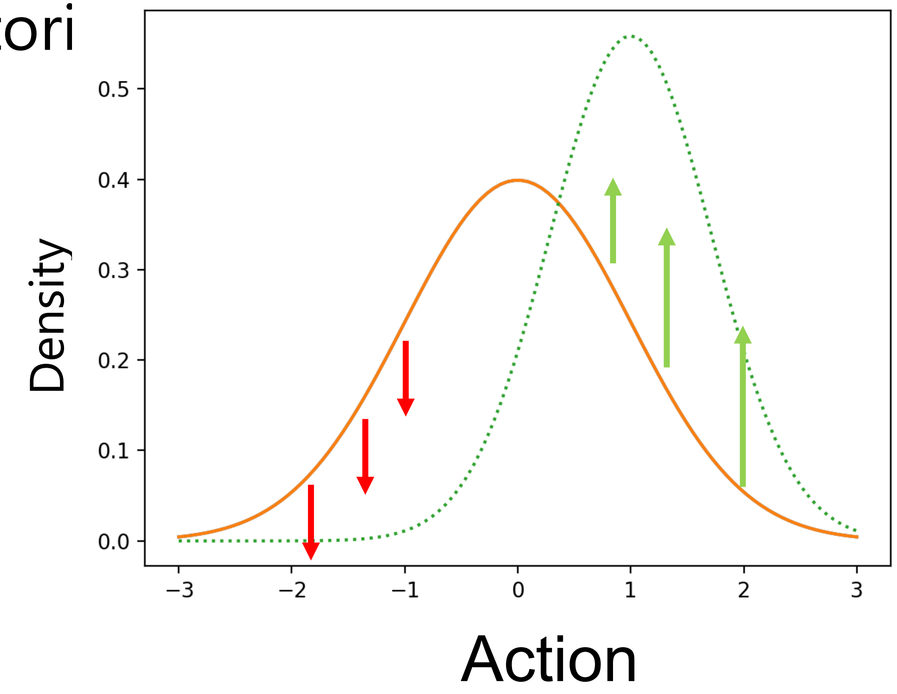
$$\approx \frac{1}{N} \sum_{i=0}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i|s_t^i) \sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)$$ (approximating using samples)

What does this mean?

$$\nabla_\theta J(\theta) = \int p_\theta(\tau)\nabla_\theta \log p_\theta(\tau) d\tau \approx \frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)$$

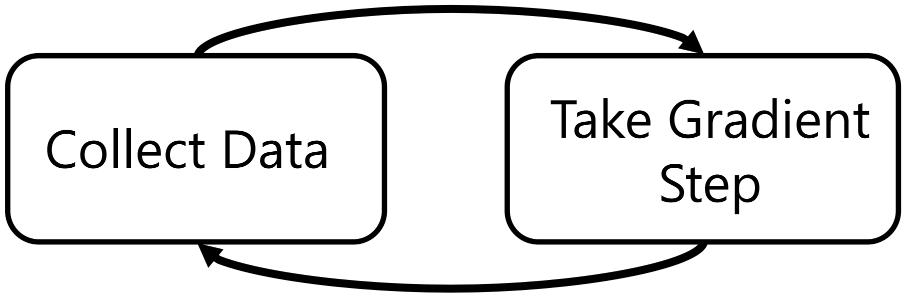
Increase the likelihood of actions in high return trajectori





Resulting Algorithm (REINFORCE)

$$\nabla_\theta J(\theta) = \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) d\tau$$



REINFORCE algorithm:

On-policy →

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run it on the robot)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right) \left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i)\right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

How is this related to supervised learning?

Reinforcement Learning

$$\nabla_\theta J(\theta) = \int p_\theta(\tau)\nabla_\theta \log p_\theta(\tau) d\tau$$

$$\approx \frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)$$

Supervised Learning

$$\max_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\log p_\theta(y|x)\right]$$

$$\approx \frac{1}{N}\sum_{i}\nabla_\theta \log p_\theta(y^i|x^i)$$

PG = select good data + increase likelihood of selected data

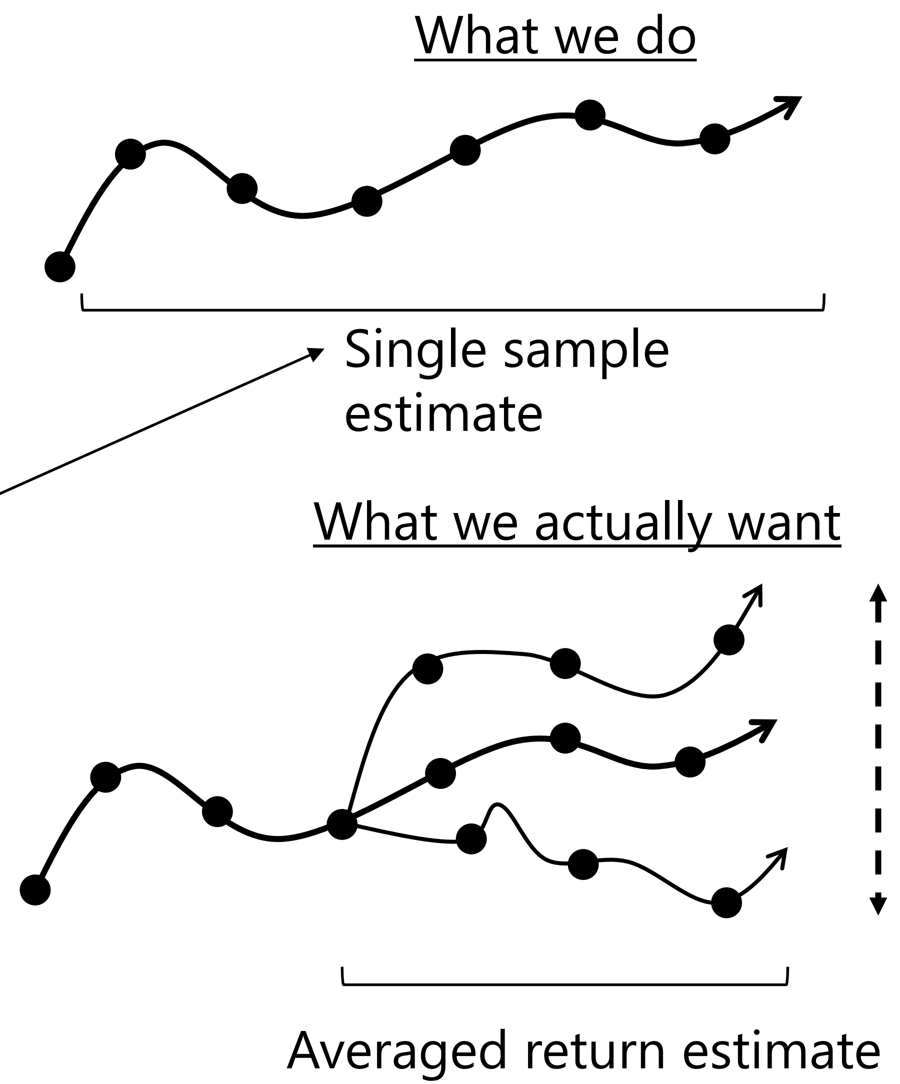
What makes policy gradient challenging?

$$\nabla_\theta J(\theta) = \int p_\theta(\tau)\nabla_\theta \log p_\theta(\tau) d\tau$$

$$\approx \frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)$$

High variance estimator!!

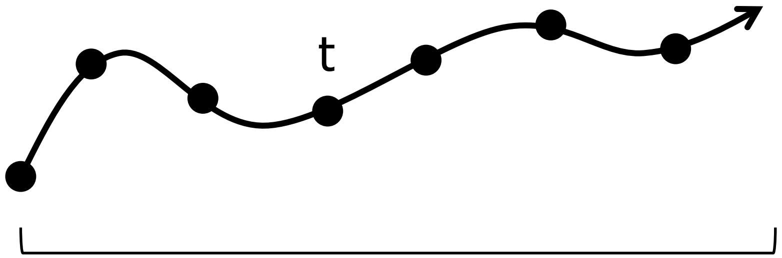
Hard to tell what matters without many samples



Variance Reduction with Causality

Idea: Trajectory returns depend on past and future, but we only care about the future, since actions cannot affect the past. Instead, consider **"return-to-go"**

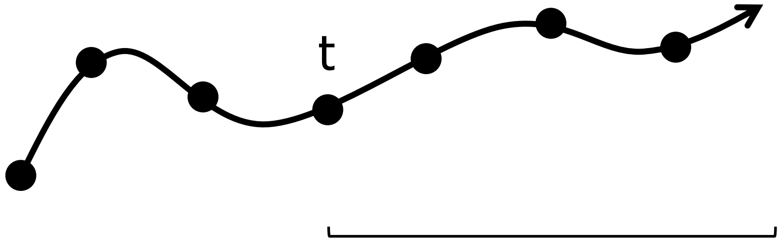
$$\approx \frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\underbrace{\sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)}_{\text{Includes } t' < t}$$



Full trajectory return

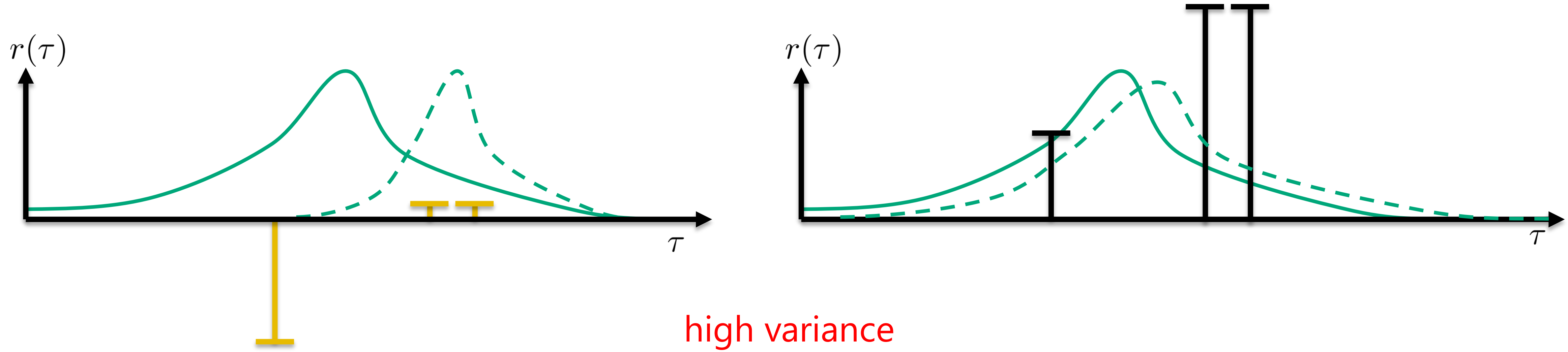
Ignore past terms ⇓

$$\frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\sum_{t'=t}^{T} r(s_t^i, a_t^i)$$



Return to go

Can we reduce variance further?



high variance

Arbitrarily uncentered, scaled returns can lead to huge variance:

→ Imagine all rewards were positive, every action would be pushed up, some more than others

→ What if instead, we pushed down some actions and pushed up some others (even if rewards are posi

Credit: Sergey Levine

Variance Reduction with a Baseline

Idea: We can reduce variance by subtracting a current state dependent function from the policy gradient return

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\left[\sum_{t'=t}^{T} r(s_{t'}^i, a_{t'}^i) - b(s_t)\right]$$

Baseline: Centers the returns, reduces variance

But does this increase bias??

Variance Reduction with a Baseline

$$\int_{\mathcal{S}} \int_{\mathcal{A}} p(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \left[\sum_{t'=t}^{T} r(s_{t'}, a_{t'}) - b(s_t) \right] ds_t \, da_t$$

$$\int_{\mathcal{S}} \int_{\mathcal{A}} p(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \left[\sum_{t'=t}^{T} r(s_{t'}, a_{t'}) \right] ds_t \, da_t - \int_{\mathcal{S}} \int_{\mathcal{A}} p(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t) \, ds_t \, da_t$$

Let us show this is 0!

Variance Reduction with a Baseline

$$\int\int p(s_t, a_t)\nabla_\theta \log \pi_\theta(a_t|s_t)\left[b(s_t)\right] ds_t da_t = \int\int p(s_t)\pi_\theta(a_t|s_t)\nabla_\theta \log \pi_\theta(a_t|s_t)\left[b(s_t)\right] ds_t da_t$$

$$= \int p(s_t)b(s_t)\int \pi_\theta(a_t|s_t)\nabla_\theta \log \pi_\theta(a_t|s_t) da_t ds_t$$

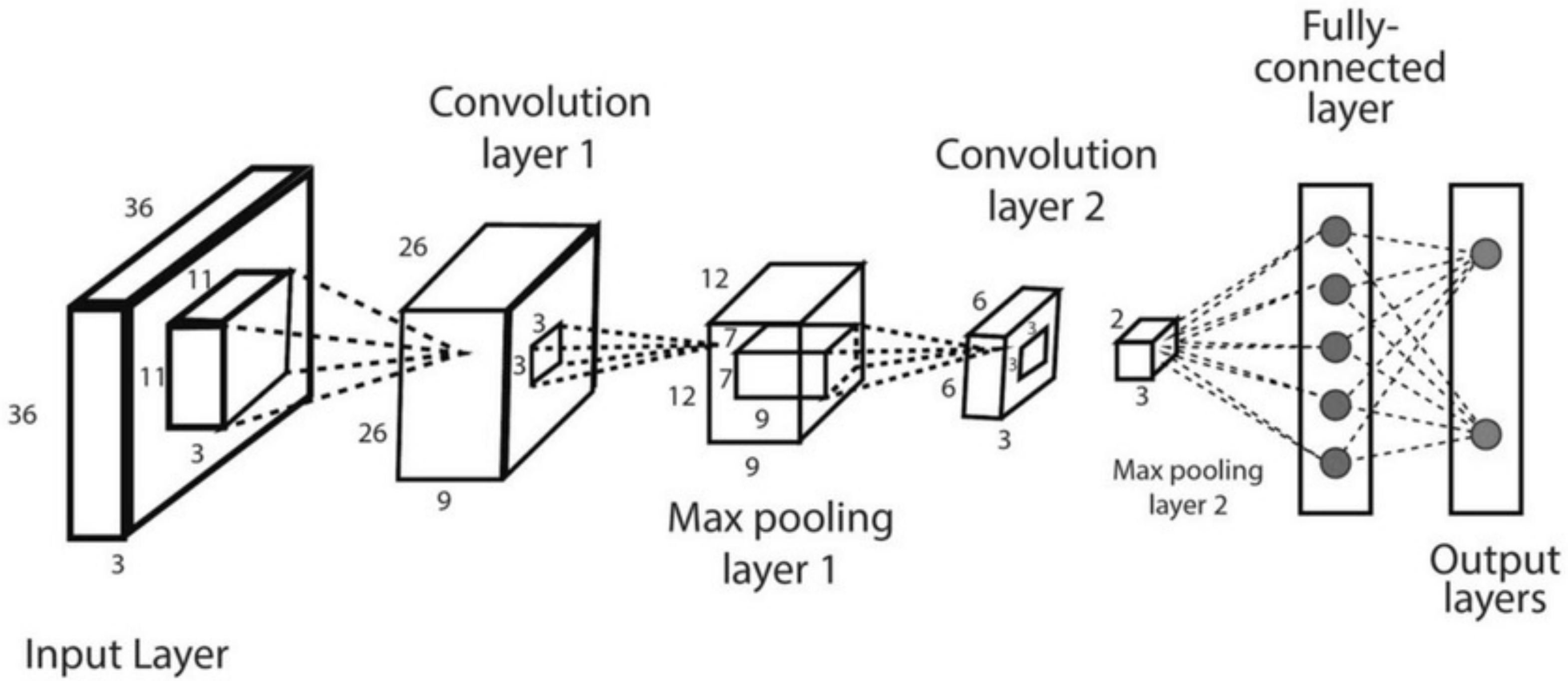
$$= \int p(s_t)b(s_t)\int \nabla_\theta \pi_\theta(a_t|s_t) da_t ds_t$$

$$= \int p(s_t)b(s_t)\nabla_\theta \int \pi_\theta(a_t|s_t) da_t ds_t = \int p(s_t)b(s_t)\nabla_\theta(1) ds_t = 0$$

Unbiased!

Learning Baselines

Baselines are typically learned as deep neural nets from $R^s \rightarrow R^1$



$$\frac{1}{N}\sum_{j=1}^{N} \|\hat{V}(s_t^j, a_t^j) - \sum_{t=1}^{H} r(s_t^j, a_t^j)\|$$

Minimize with Monte-carlo regression at every iteration, club with policy loss

$$A(s_t, a_t) = \sum_{t'=t}^{T} r(s'_t, a'_t) - V(s_t)$$

Allows us to define advantages

Lecture Outline

Reinforcement Learning: Motivation

↓

Imitation Learning

↓

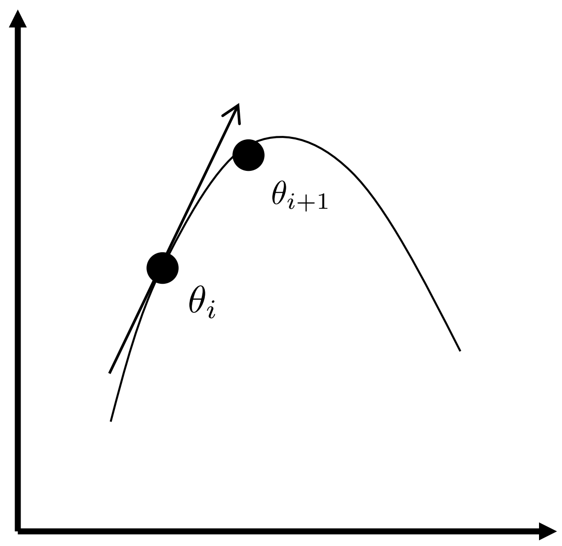
Policy Gradient and Beyond

Take a deeper look at REINFORCE

$$\nabla_\theta J(\theta) = \int p_\theta(\tau)\nabla_\theta \log p_\theta(\tau) d\tau \approx \frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)$$

Gradient descent is steepest descent on linear approximation under the Euclidean metric

$$\max_\theta \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} r(s_t, a_t)\right]$$
$$= J(\theta)$$



Take a deeper look at REINFORCE

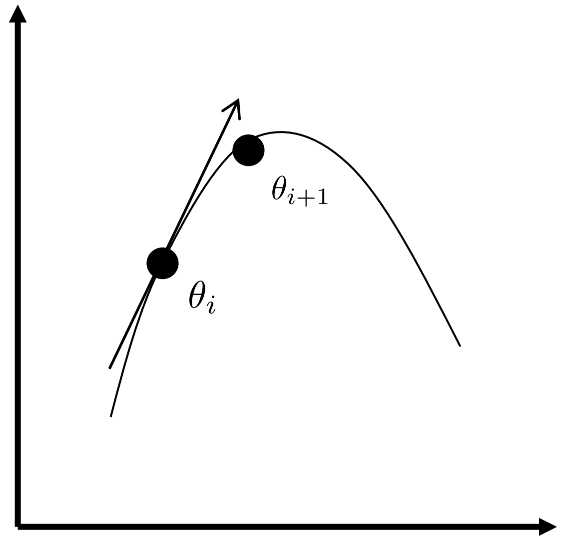
$$\nabla_\theta J(\theta) = \int p_\theta(\tau)\nabla_\theta \log p_\theta(\tau) d\tau \approx \frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)$$

Gradient descent is steepest descent on linear approximation under the Euclidean metric

$$\max \;\; J(\theta_i) + \nabla_\theta J(\theta)|_{\theta=\theta_i}(\theta - \theta_i)$$ Linear approximation

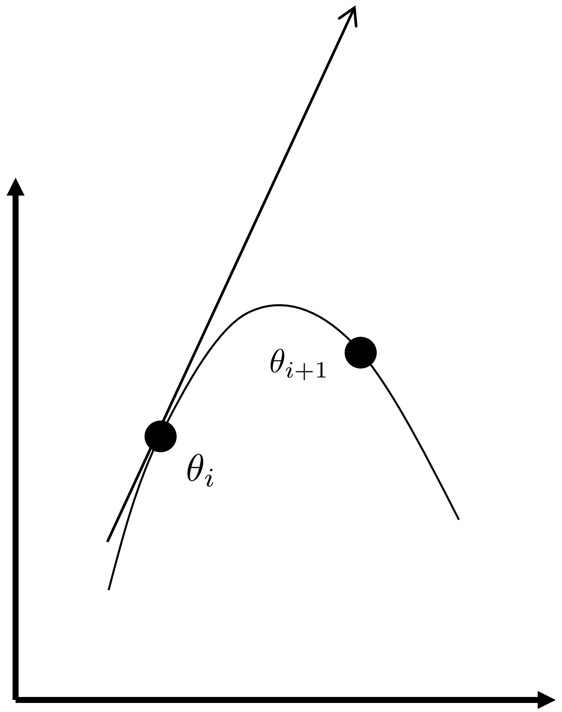
$$(\theta - \theta_i)^T(\theta - \theta_i) \leq \epsilon$$ Quadratic Constraint

$$\theta = \theta_i + \alpha\nabla_\theta J(\theta)|_{\theta=\theta_i}$$



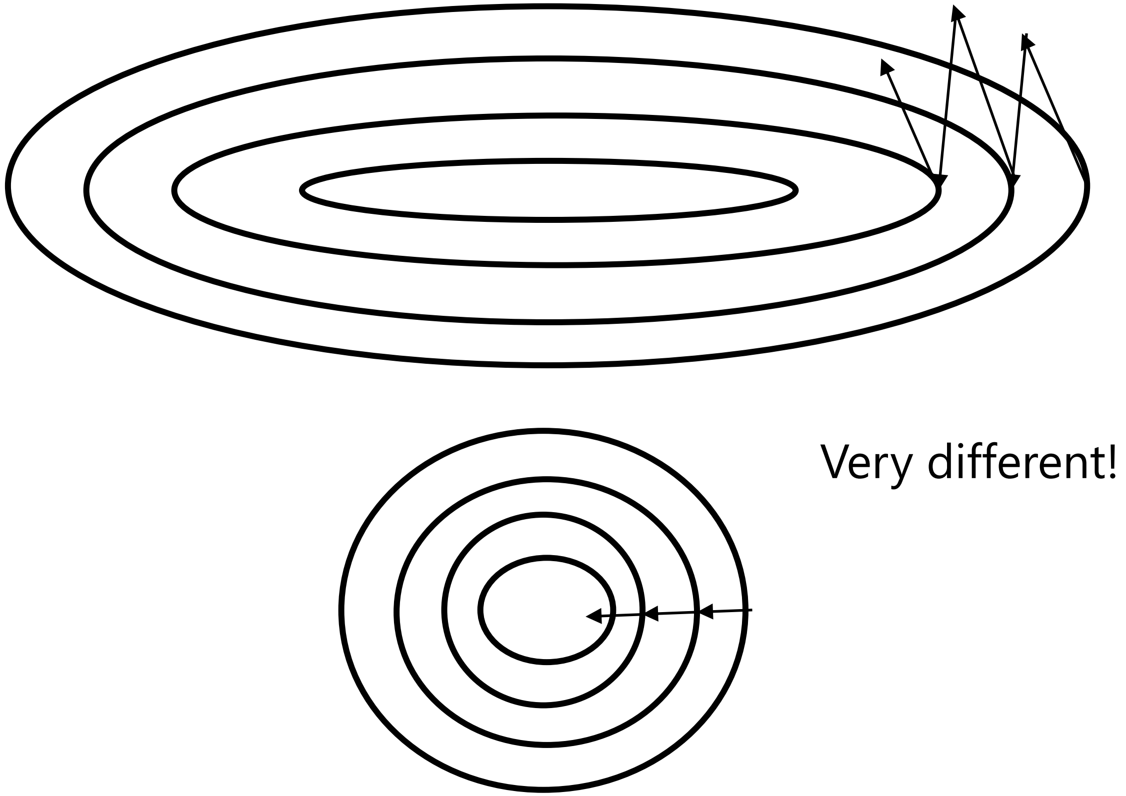
When might this fail?

<u>Large step sizes may cause collapse</u>



Must use very small step sizes, slow!

<u>Sensitive to Policy Parameterization</u>



Can struggle for a deep neural network!

Parameterization dependence of PG

Sensitive to Policy Parameterization

$$L(\theta) = \theta_1 + \theta_2$$

$$L(\phi) = \phi_1^{0.5} + \phi_2^{-1}$$

$$\phi_1 = \theta_1^2$$

$$\phi_2 = \theta_2^{-1}$$

$$\nabla_{\theta_1} L = 1$$

$$\nabla_{\theta_2} L = 1$$

$$\nabla_{\phi_1} L = 0.5\phi_1^{-0.5} = 0.5\theta_1^{-1}$$

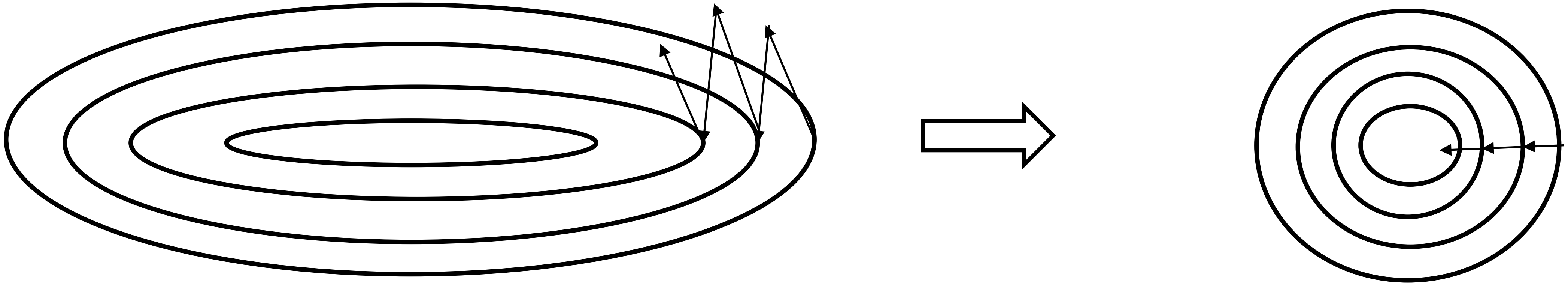
$$\nabla_{\phi_2} L = -\phi_2^{-2} = -\theta_2^2$$

↔ Not covariant!

Modified Constraint on Policy Gradient

$$\max \quad J(\theta_i) + \nabla_\theta J(\theta)|_{\theta=\theta_i}(\theta - \theta_i)$$
$$(\theta - \theta_i)^T(\theta - \theta_i) \leq \epsilon$$

$$\max \quad J(\theta_i) + \nabla_\theta J(\theta)|_{\theta=\theta_i}(\theta - \theta_i)$$
$$(\theta - \theta_i)^T G(\theta - \theta_i) \leq \epsilon$$



$$\theta_{i+1} = \theta_i + \alpha G^{-1} \nabla_\theta J(\theta)|_{\theta=\theta_i}$$

Rescales according to G^{-1}

Adaptive choice of G can avoid sensitivity to policy parameterization!

Covariant Policy Gradient Updates

$$\max \; J(\theta_i) + \nabla_\theta J(\theta)|_{\theta=\theta_i}(\theta - \theta_i)$$
$$(\theta - \theta_i)^T G(\theta - \theta_i) \leq \epsilon$$

What should G be?

$$\max \; J(\theta_i) + \nabla_\theta J(\theta)|_{\theta=\theta_i}(\theta - \theta_i)$$
$$D_{\mathrm{KL}}(\pi_\theta || \pi_{\theta_i}) \leq \epsilon$$

Let us use the constraint as KL divergence on the policy
(2nd order Taylor expansion)

Measures functional distance, not parameter distance

Resulting "Natural" Policy Gradient

$$\max \;\; J(\theta_i) + \nabla_\theta J(\theta)|_{\theta=\theta_i}(\theta - \theta_i)$$

$$D_{\mathrm{KL}}(\pi_\theta || \pi_{\theta_i}) \leq \epsilon$$

2nd order approximation of KL → Fisher Information Metric

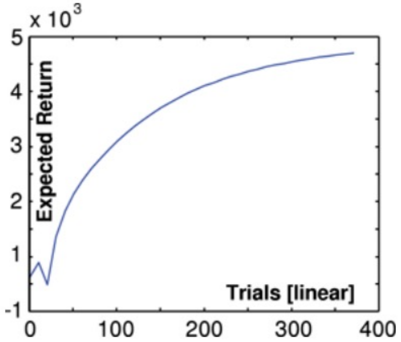
$$F = \mathbb{E}_{\pi_\theta}\left[(\nabla_\theta \log \pi_\theta)(\nabla_\theta \log \pi_\theta)^T\right]$$

$$\max \;\; J(\theta_i) + \nabla_\theta J(\theta)|_{\theta=\theta_i}(\theta - \theta_i)$$

$$(\theta - \theta_i)^T F (\theta - \theta_i) \leq \epsilon$$

Resulting update $$\theta_{i+1} = \theta_i + \alpha F^{-1} \nabla_\theta J(\theta)|_{\theta=\theta_i}$$ Covariant to parameterization

Natural Policy Gradient in Action

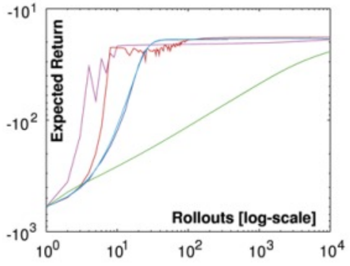


(a) Performance.

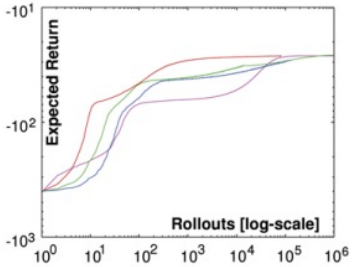
(b) Imitation learning.

(c) Initial reproduction.

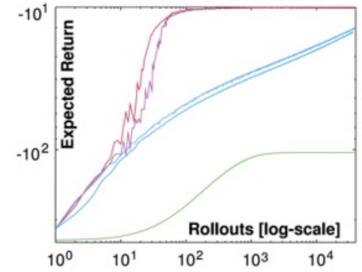
(d) After reinforcement learning.



(b) Minimum motor command with motor primitives



(c) Passing through a point with splines



(d) Passing through a point with motor primitives

Finite Difference Gradient
Vanilla Policy Gradient with constant baseline
Vanilla Policy Gradient with time-variant baseline
Episodic Natural Actor-Critic with single offset basis functions
Episodic Natural Actor-Critic with time-variant offset basis functions

Peters, Schaal '08

Recap: Course Overview