Reinforcement Learning

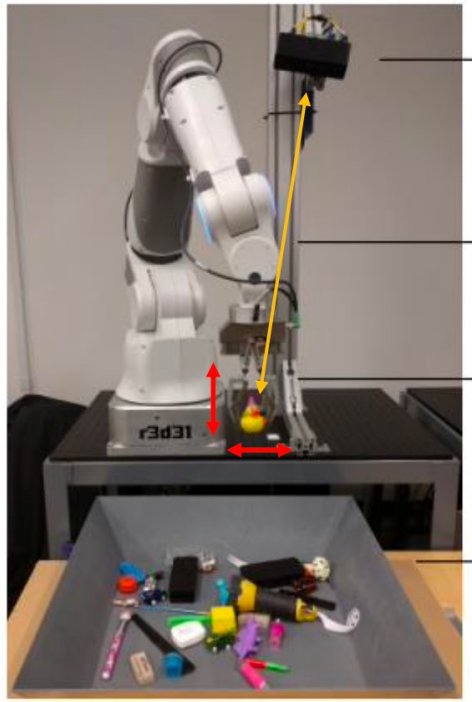
Computer Engineering Department
Sharif University of Technology

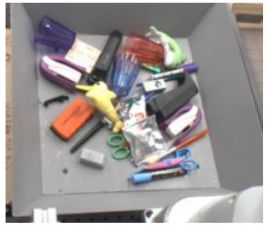
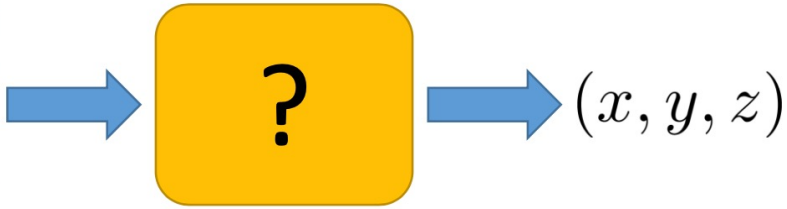
Mohammad Hossein Rohban, Ph.D.

Spring 2026

Courtesy: Some slides are adopted from CS 285 Berkeley, and CS 234 Stanford, and Pieter Abbeel's compact series on RL.

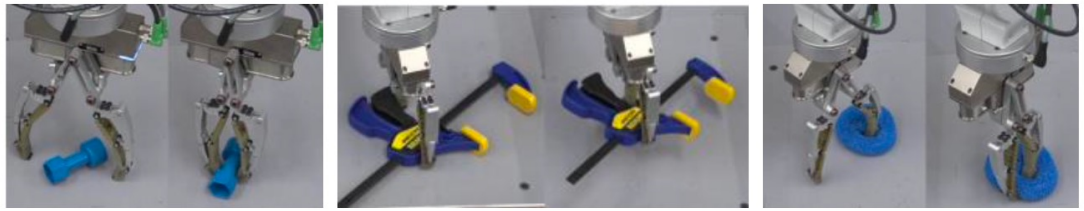
Motivation



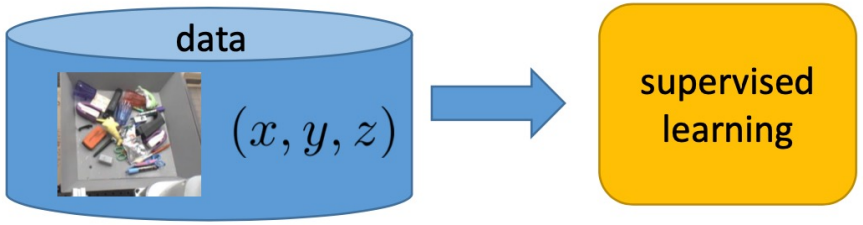
Option 1:

Understand the problem, design a solution



Option 2:

Set it up as a machine learning problem

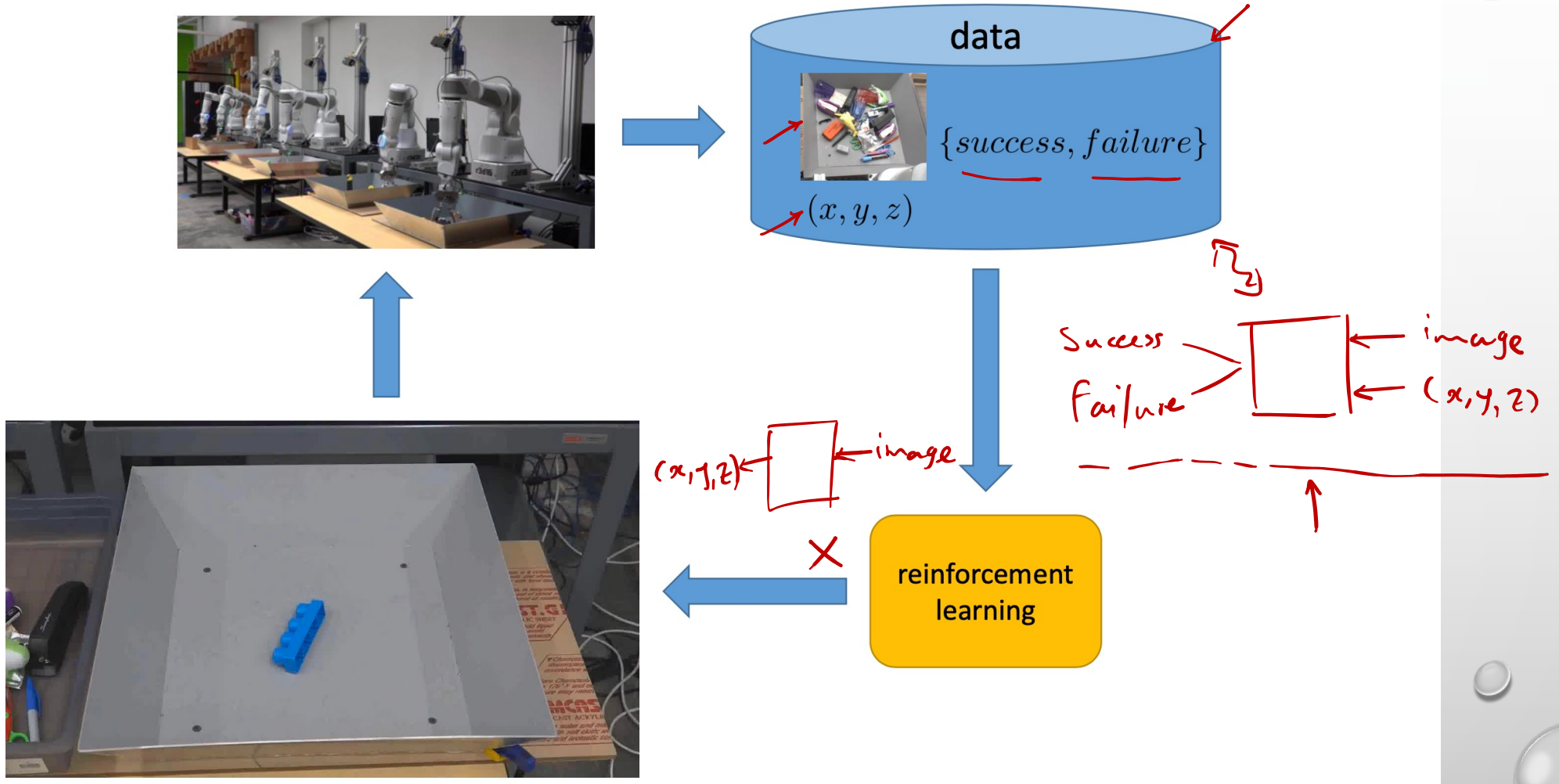


Courtesy: CS 285 course, Berkeley

Motivation (cont.)

Courtesy: CS 285 course, Berkeley

Motivation (cont.)

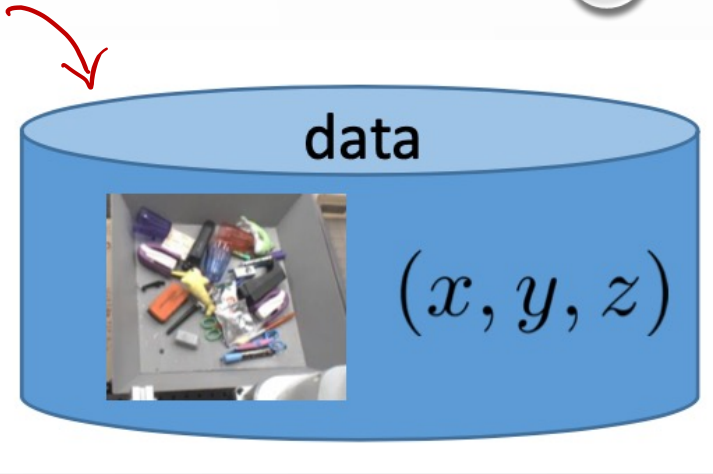


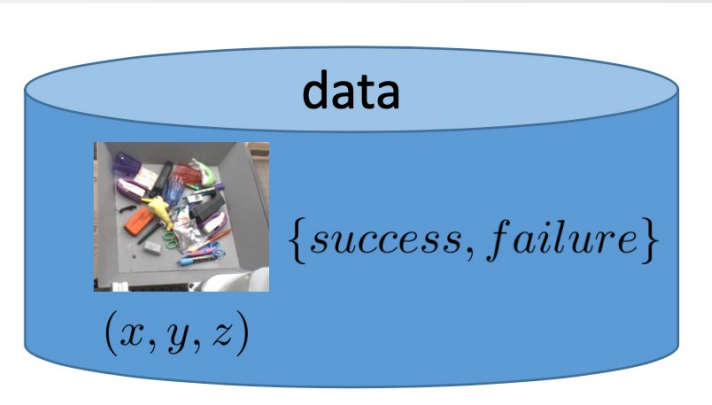
Courtesy: CS 285 course, Berkeley

Motivation (cont.)

Credit Assignment

- Supervised learning:
  - Ground truth is known in advance.
  - Training data are usually static and iid.
- Reinforcement learning:
  - The best action (**policy**) is usually unknown a priori.
  - Sequence of actions is needed.
  - A series of trial and error (search) is performed.
    - Usually delayed **reward** shows goodness of the trial.
  - Data is dynamic (**exploration**) and non-iid.





What is Reinforcement Learning?

$\max_\theta \mathbb{E}_{\pi_\theta}\left\{\sum_{t=1}^{T} r_t\right\}$

supervised learning

Model

computer program

[object label]

input: $\mathbf{x}$

output: $\mathbf{y}$

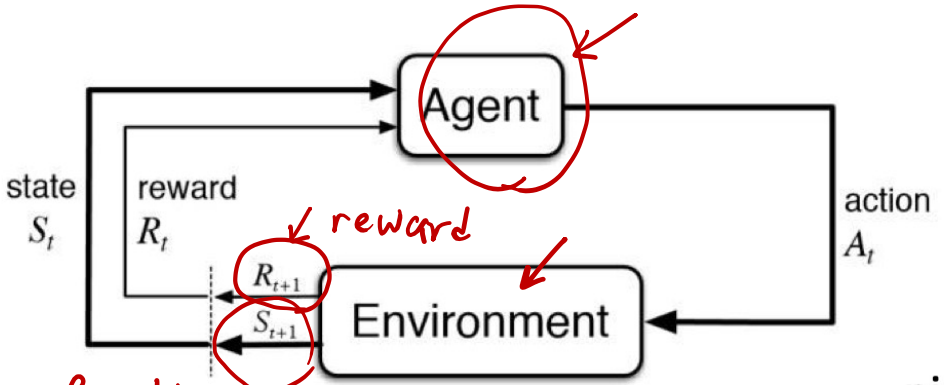
data: $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}$

goal: $f_\theta(\mathbf{x}_i) \approx \mathbf{y}_i$

someone gives this to you

$\min_\theta \sum_i \ell(f_\theta(x_i), y_i)$

reinforcement learning



input: $\mathbf{s}_t$ *at each time step*

output: $\mathbf{a}_t$ *at each time step*

data: $(\mathbf{s}_1, \mathbf{a}_1, r_1, \mathbf{s}_2, \ldots, \mathbf{s}_T, \mathbf{a}_T, r_T)$

trajectory

goal: learn π_θ : $\mathbf{s}_t \rightarrow \mathbf{a}_t$

to maximize $\sum_t r_t$

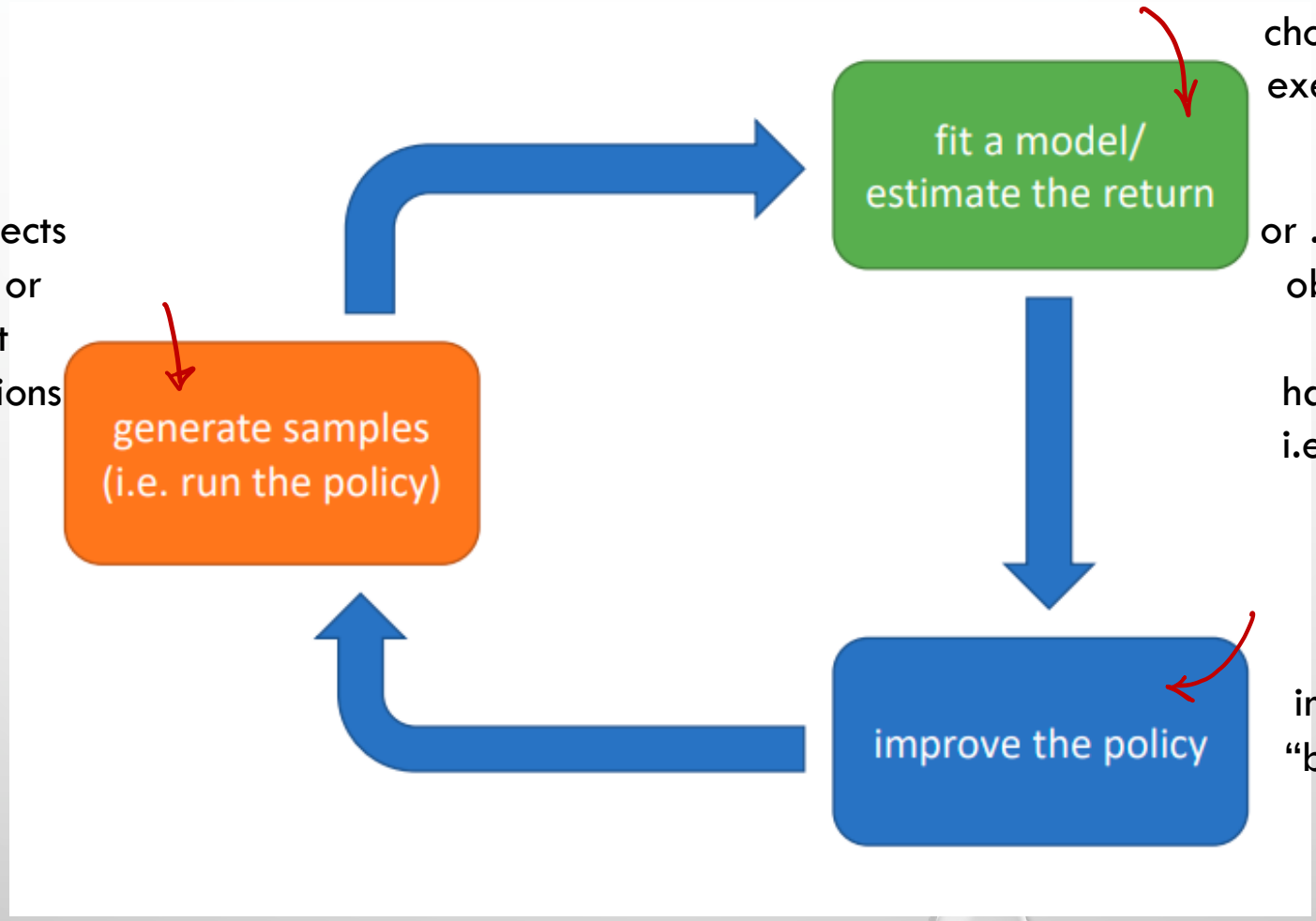
return

pick your own actions

$\sum_{t=1}^{T} \gamma^t r_t$ $\gamma < 1$

Courtesy: CS 285 course, Berkeley

The Anatomy of Reinforcement Learning



how well was our choice of (x, y, z); execute gripping!

gripping objects on random or else "best guess" locations

or … when I grip some object on a specific location, what happens physically; i.e. how its (x, y, z)'s change?

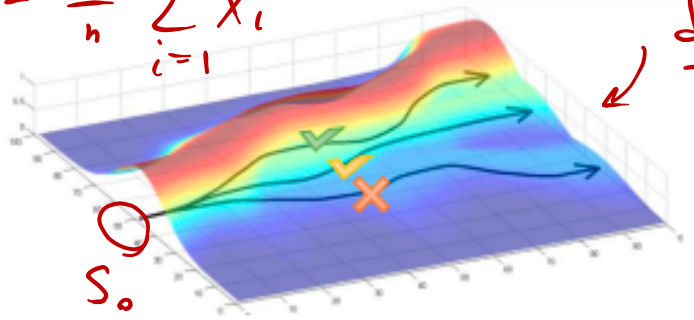
improve our "best guess"

Courtesy: CS 285 course, Berkeley

A Simple Example

$$\mathbb{E}(X) = \int x \cdot f_X(x)\, dx = \frac{1}{n}\sum_{i=1}^{n} X_i$$

density

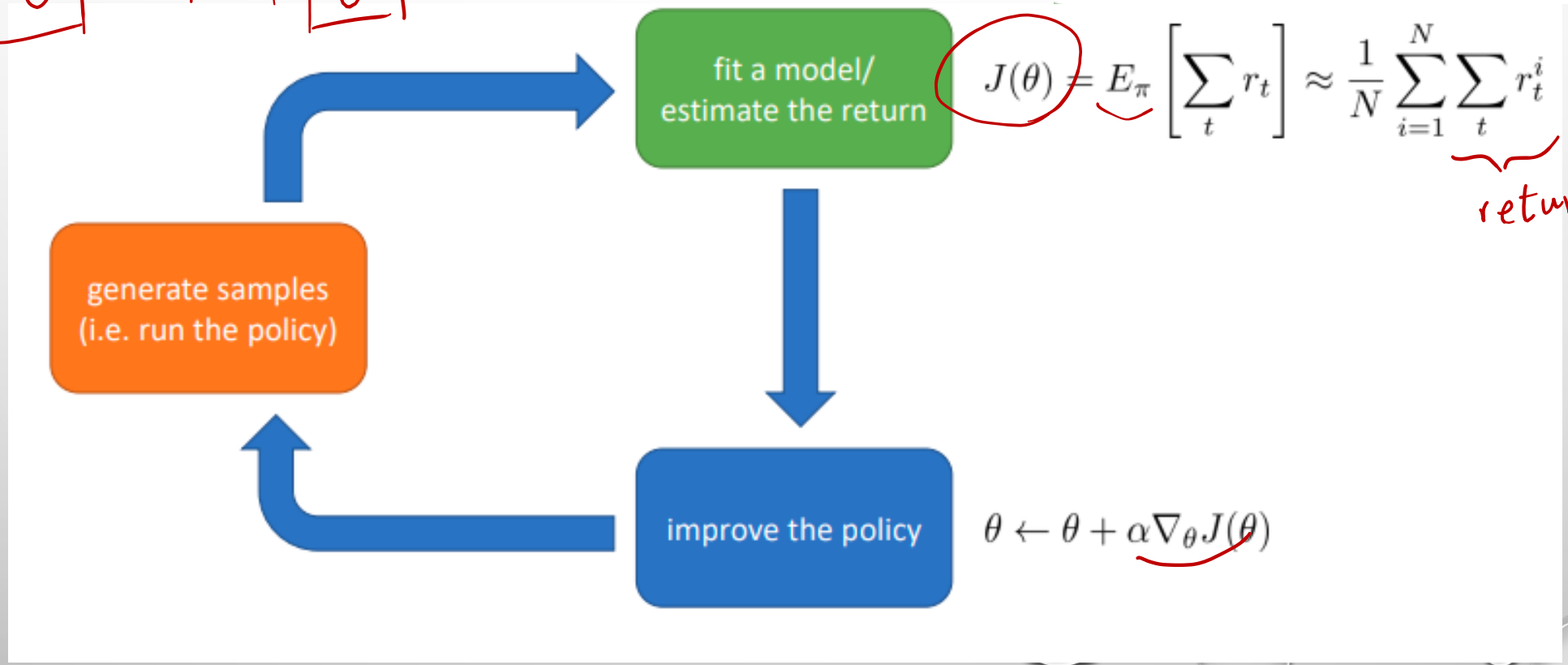


s_0

π

$s_t \rightarrow \boxed{\theta} \rightarrow a_t$

$s_0 \rightarrow \boxed{\theta} \rightarrow a_1 \quad s_1 \rightarrow \boxed{\theta} \rightarrow a_2 \cdots \rightarrow a_T$



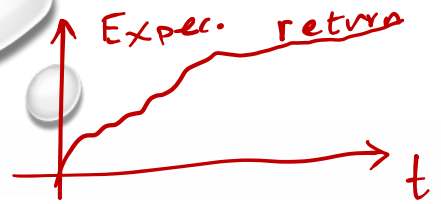
$$J(\theta) = E_\pi\left[\sum_t r_t\right] \approx \frac{1}{N}\sum_{i=1}^{N}\sum_t r_t^i$$

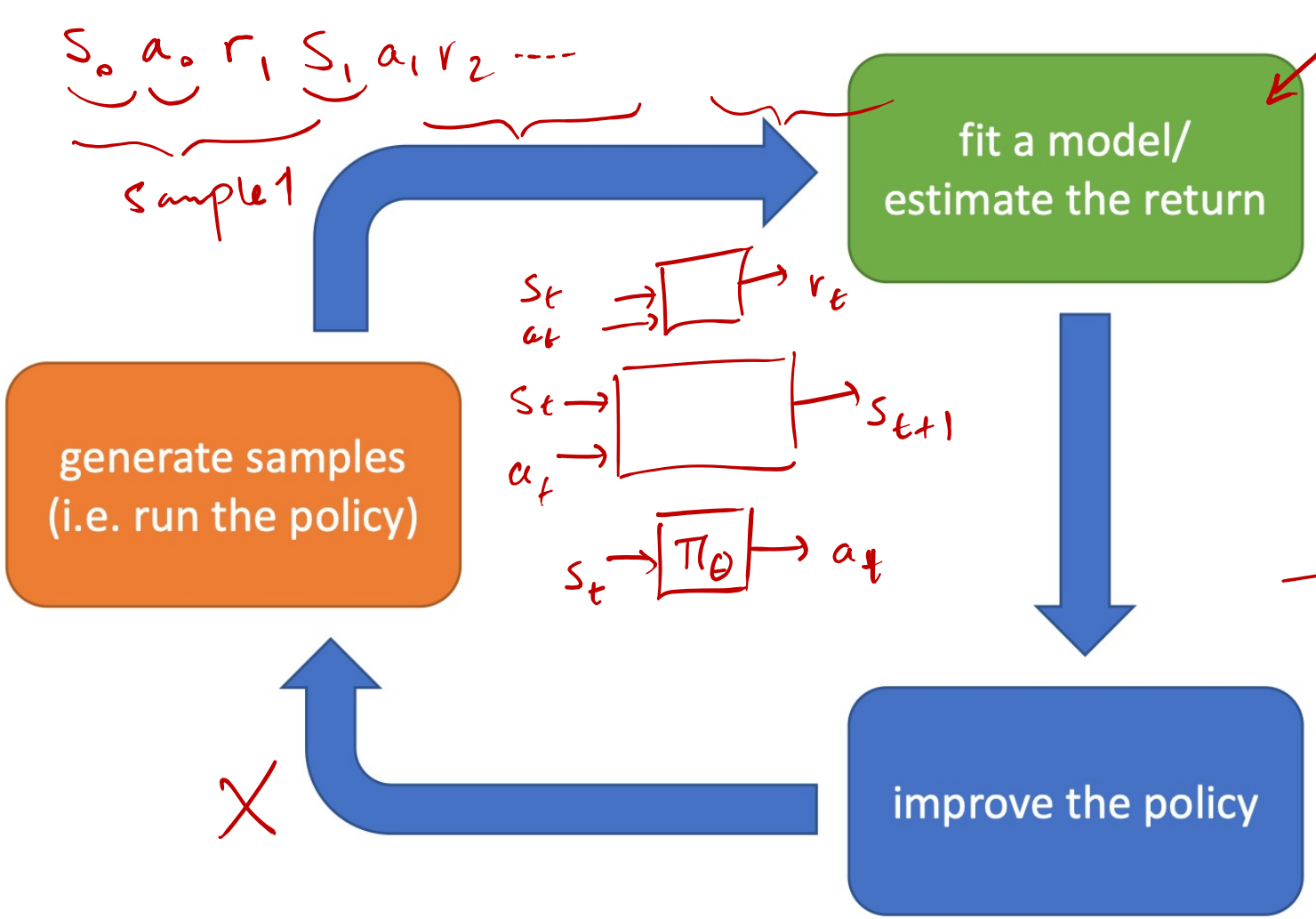
return i

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

Courtesy: CS 285 course, Berkeley

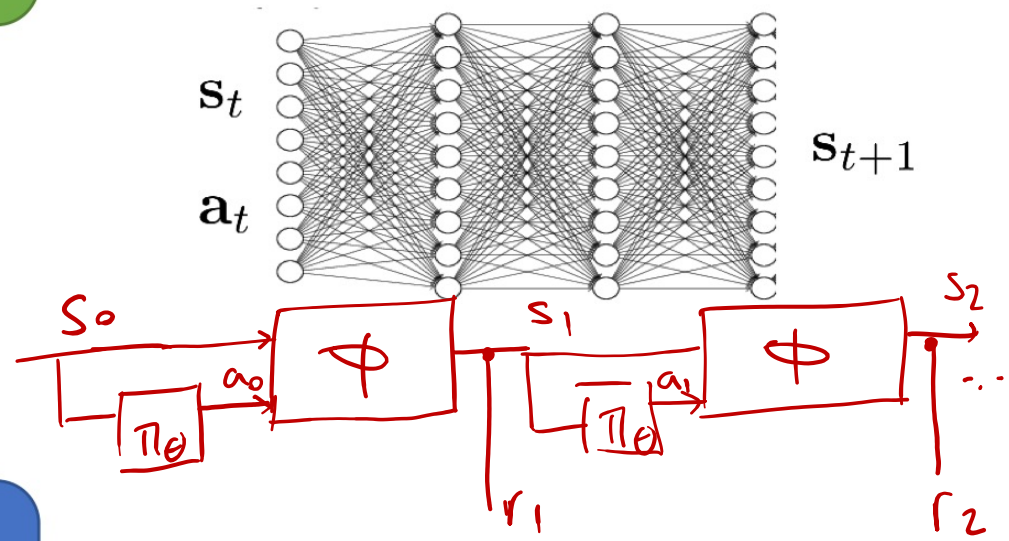
Another Example





World Model

learn f_ϕ such that $\mathbf{s}_{t+1} \approx f_\phi(\mathbf{s}_t, \mathbf{a}_t)$



backprop through f_ϕ and r to train $\pi_\theta(\mathbf{s}_t) = \mathbf{a}_t$

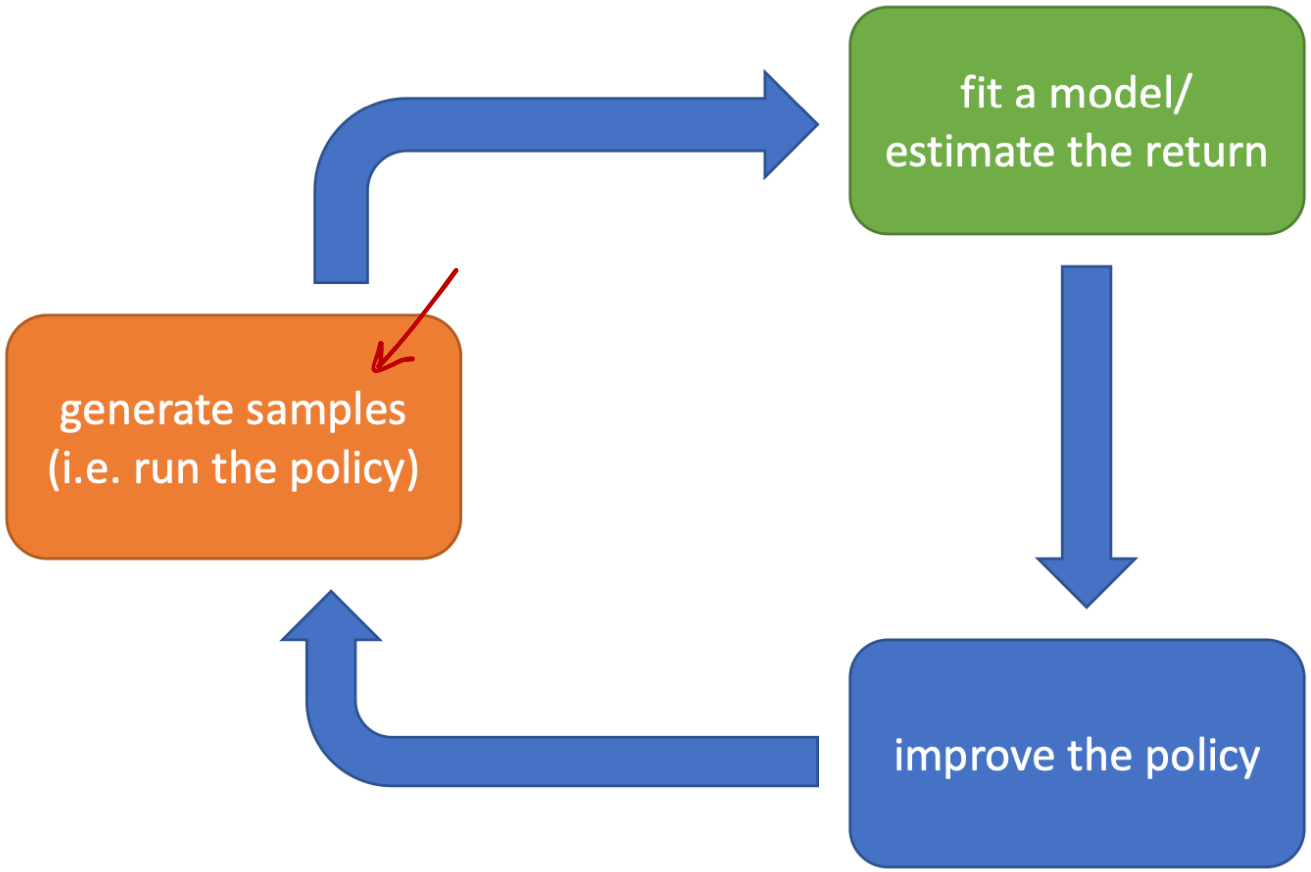
$\max_\theta \sum r_t$

Courtesy: CS 285 course, Berkeley

Which parts are expensive?

$$J(\theta) = E_\pi \left[\sum_t r_t \right] \approx \frac{1}{N} \sum_{i=1}^{N} \sum_t r_t^i$$

trivial, fast



real robot/car/power grid/whatever:
1x real time, until we invent time travel

MuJoCo simulator:
up to 10000x real time

$$\text{learn } \mathbf{s}_{t+1} \approx f_\phi(\mathbf{s}_t, \mathbf{a}_t)$$

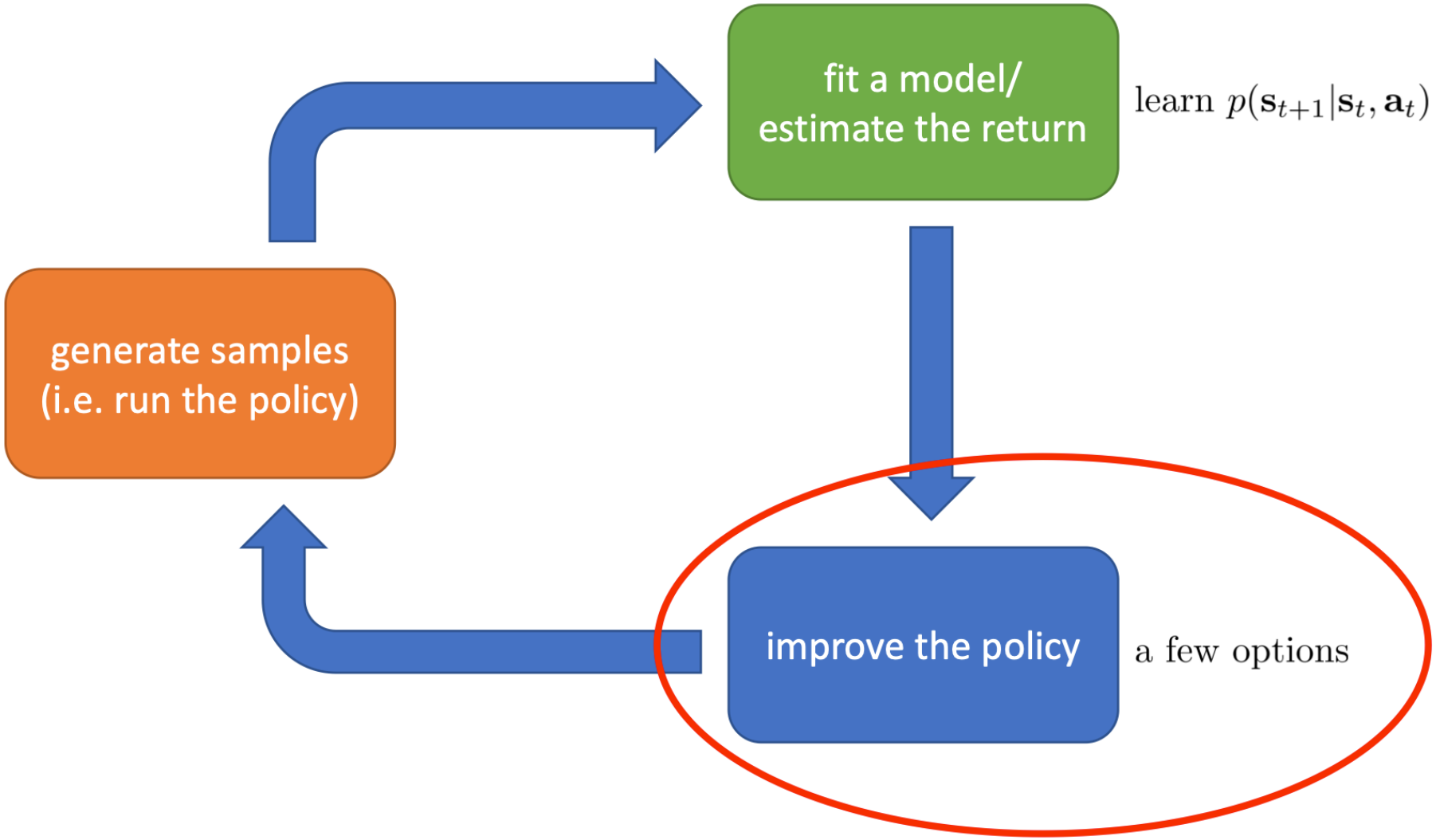
expensive

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

backprop through f_ϕ and r to train $\pi_\theta(\mathbf{s}_t) = \mathbf{a}_t$

Courtesy: CS 285 course, Berkeley

Model-based RL



Courtesy: CS 285 course, Berkeley

Value-based RL

$$\pi_{new}(s) = \arg\max_a Q^{\pi}(s, a)$$

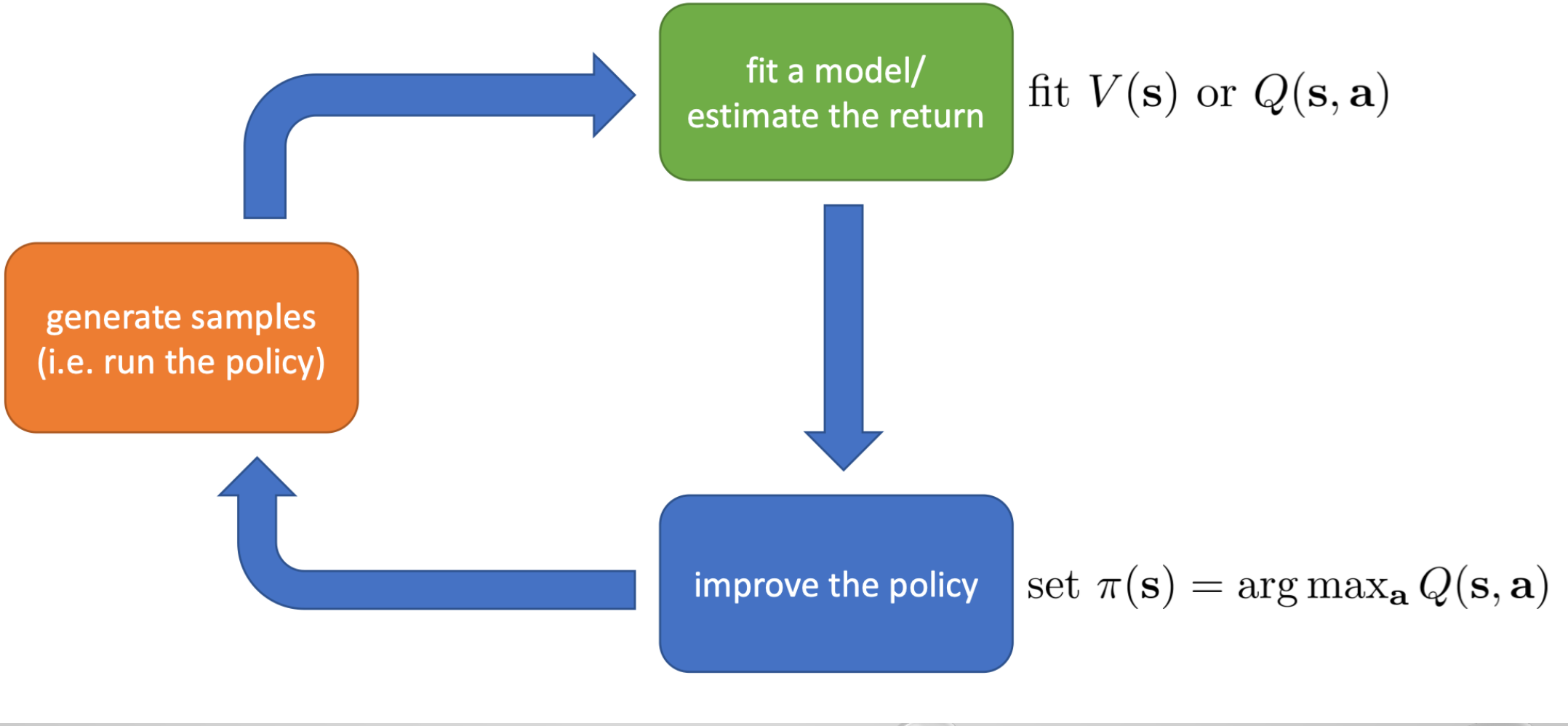
$Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t\right]$: total reward from taking $\mathbf{a}_t$ in $\mathbf{s}_t$

$V^{\pi}(\mathbf{s}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t\right]$: total reward from $\mathbf{s}_t$

$$V^{\pi}(\mathbf{s}_t) = E_{\mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t)}[Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t)]$$

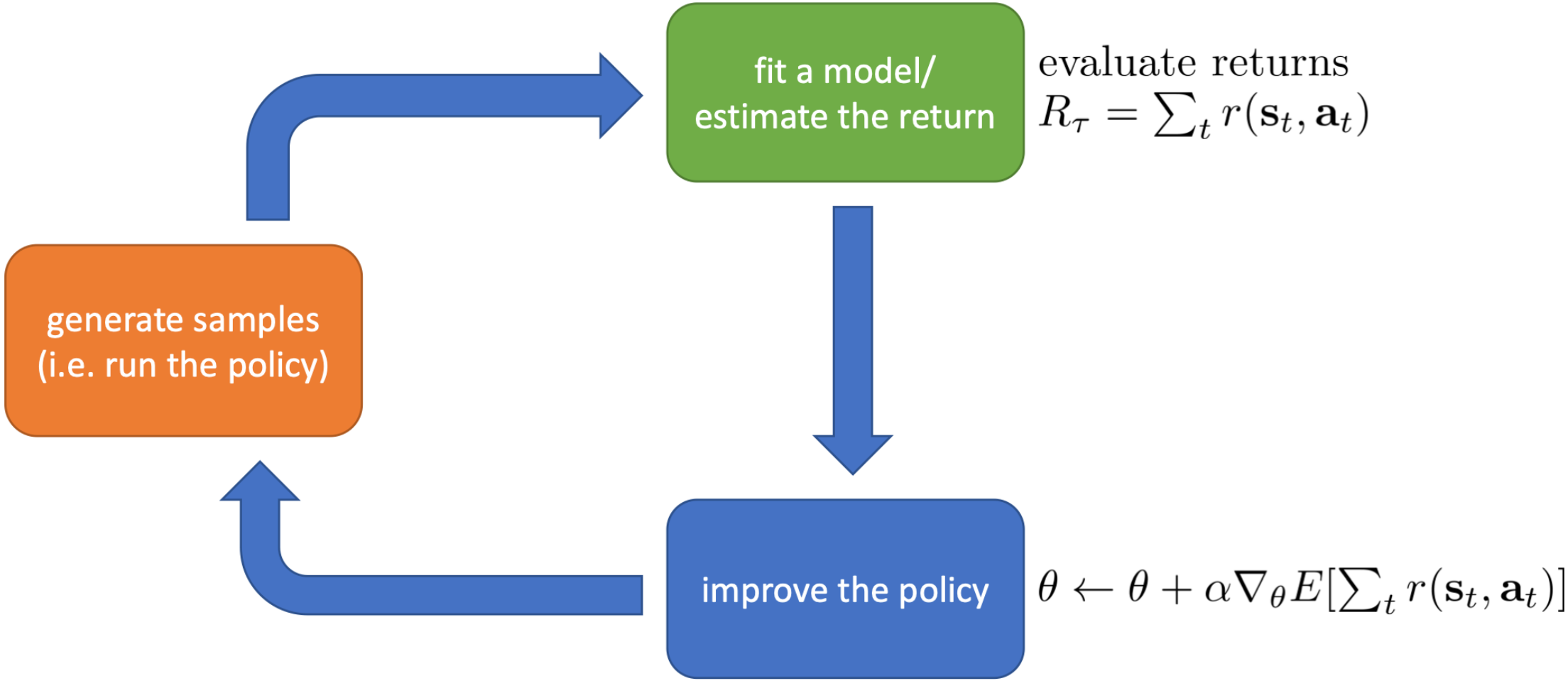
$E_{\mathbf{s}_1 \sim p(\mathbf{s}_1)}[V^{\pi}(\mathbf{s}_1)]$ is the RL objective!

Value-based RL (cont.)



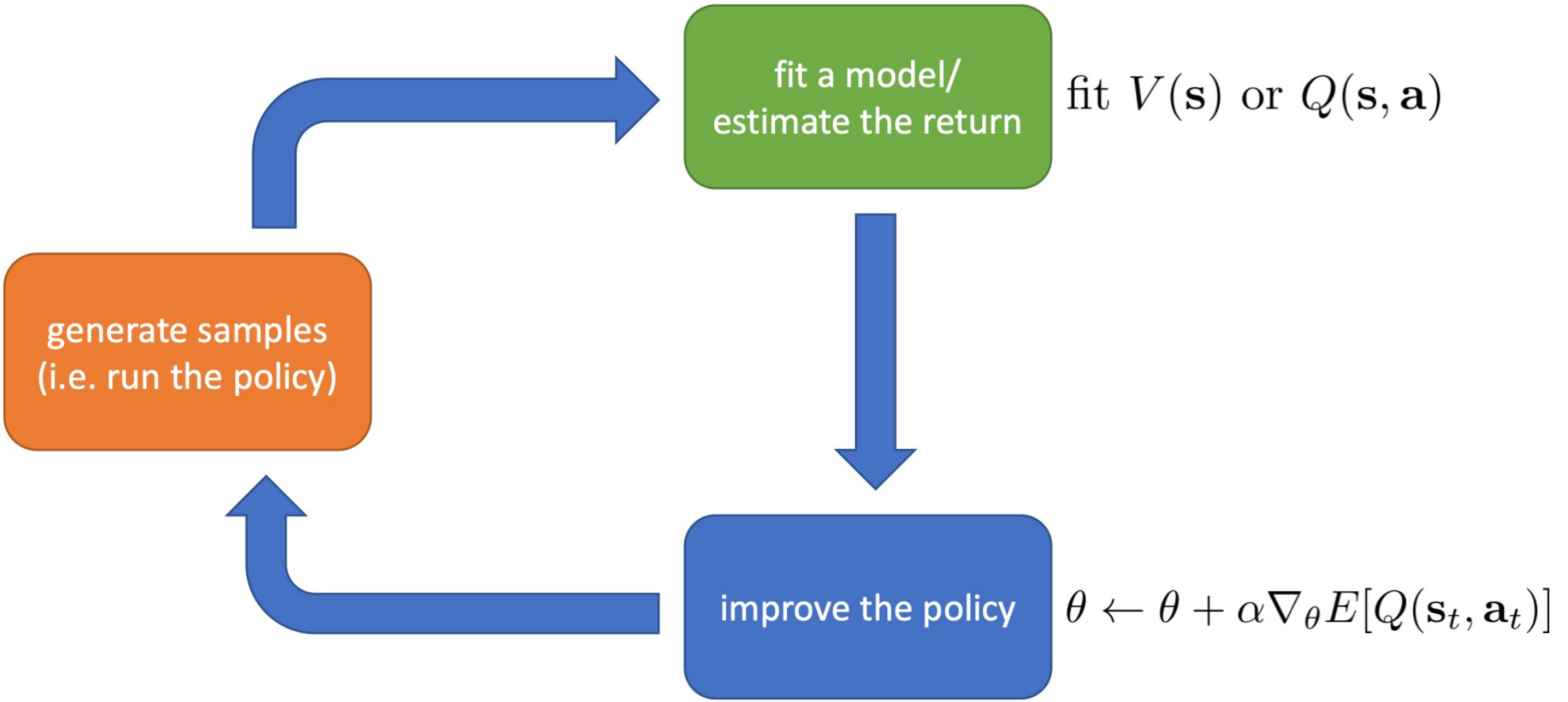
Courtesy: CS 285 course, Berkeley

Direct Policy Gradient



Courtesy: CS 285 course, Berkeley

Actor-critic: value functions + policy gradients



Courtesy: CS 285 course, Berkeley

Where do rewards come form?

- An expert gives us the reward
- Learning from demonstrations
  - Directly copying observed behavior
  - Inferring rewards from observed behavior (inverse reinforcement learning)

Motivation (cont.)

	AI Planning	SL	UL	RL	IL
Optimization	X			X	X
Learns from experience		X	X	X	X
Generalization	X	X	X	X	X
Delayed Consequences	X			X	X
Exploration				X	

- SL = supervised learning; UL = unsupervised learning; RL = reinforcement learning; IL = imitation learning
- Imitation learning typically assumes input demonstrations of good policies
- IL reduces RL to SL. IL + RL is promising area

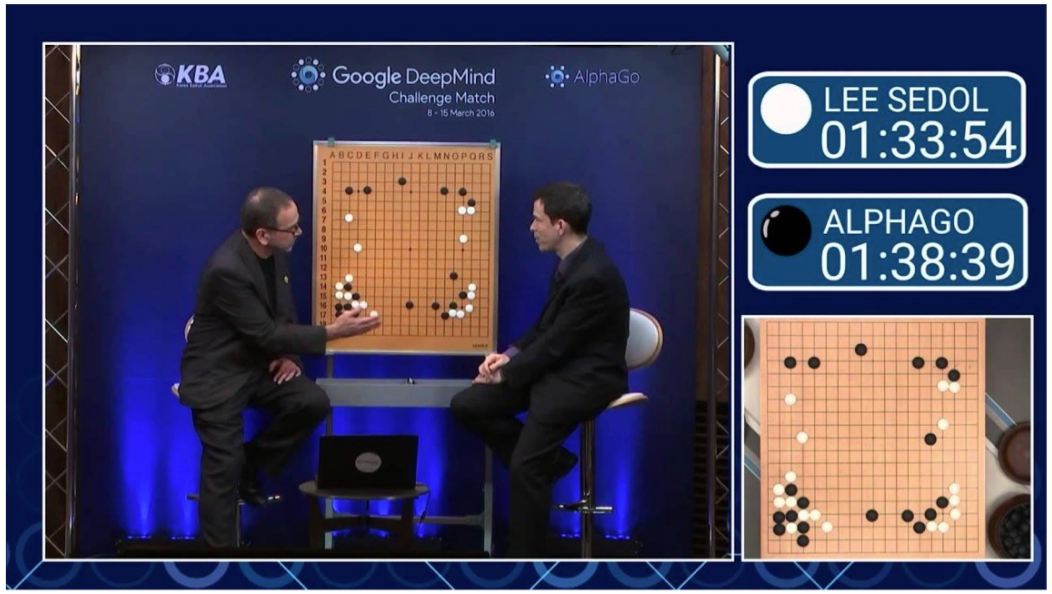
Courtesy: CS 234 course, Stanford

Planning vs learning

- Two fundamental problems in sequential decision making
  - Reinforcement learning:
    - The environment is initially unknown
    - The agent interacts with the environment
    - The agent improves its policy
  - Planning:
    - A model of the environment is known
    - The agent performs computations with its model (without any external interaction)
    - The agent improves its policy
    - a.k.a. deliberation, reasoning, introspection, pondering, thought, search

Why should we study deep reinforcement learning?

Impressive because no person had thought of it!



"Move 37" in Lee Sedol AlphaGo match: reinforcement learning "discovers" a move that surprises everyone

Impressive because it looks like something a person might draw!

Courtesy: CS 285 course, Berkeley

Data-driven AI vs. RL

Data-Driven AI

All about using **data**

+ learns about the real world from data

- doesn't try to do **better** than the data

Reinforcement Learning

All about **optimization**

+ optimizes a goal with emergent behavior

- but need to figure out how to use at scale!

Data without **optimization** doesn't allow us to solve new problems in new ways

Courtesy: CS 285 course, Berkeley

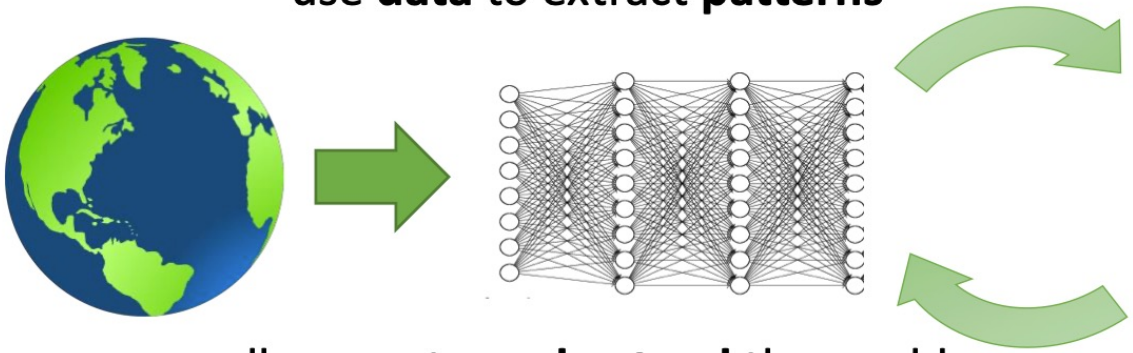
A Bitter Lesson (Richard Sutton)

"We have to learn the bitter lesson that building in how we think we think does not work in the long run. The two methods that seem to scale arbitrarily … are learning and search

http://www.incompleteideas.net/IncIdeas/BitterLesson.html

Learning

use **data** to extract **patterns**



allows us to **understand** the world

Search

use **computation** to extract **inferences**

optimization

some optimization process that uses (typically iterative) computation to make rational decisions

leverages that **understanding** for **emergence**

Data without **optimization** doesn't allow us to solve new problems in new ways

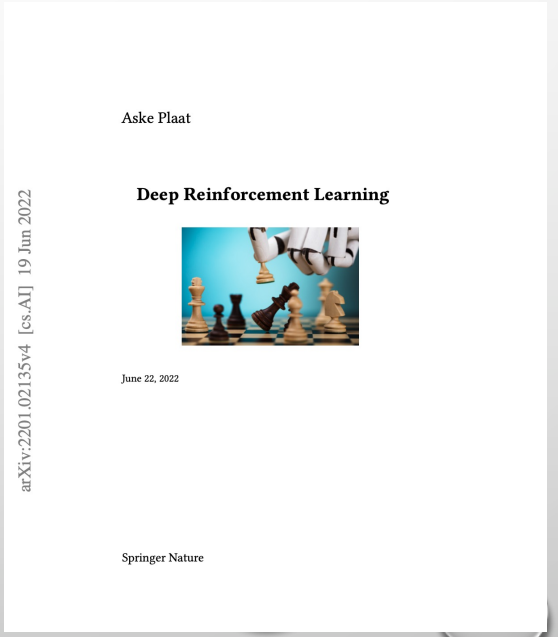
Optimization without **data** is hard to apply to the real world outside of simulators

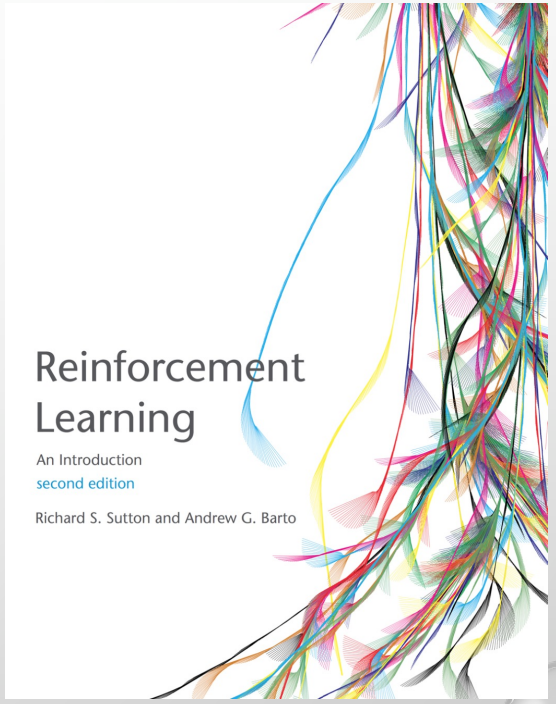
Superintelligence

- The models are trained based on human annotations and preferences.
  - Can they get smarter than humans?

References

- Reinforcement Learning: An Introduction by R. Sutton and A. Barto, 2nd Edition, 2020.
- Deep Reinforcement Learning by A. Plaat, 2022.
- Original papers of some methods.





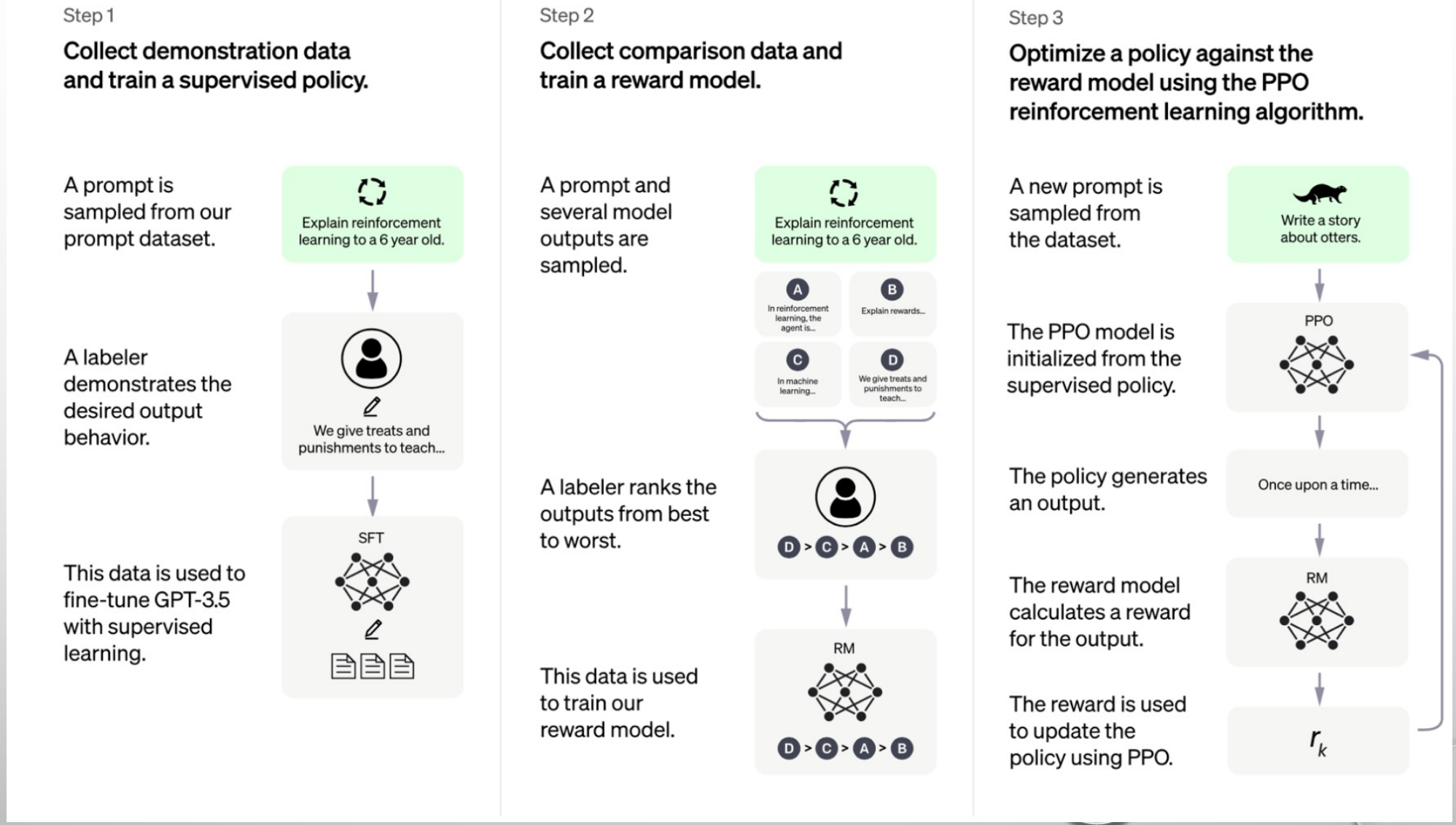
Teaching Assistants

- Ali Najar (Head TA)
- Pouya Toroghi (Head TA)

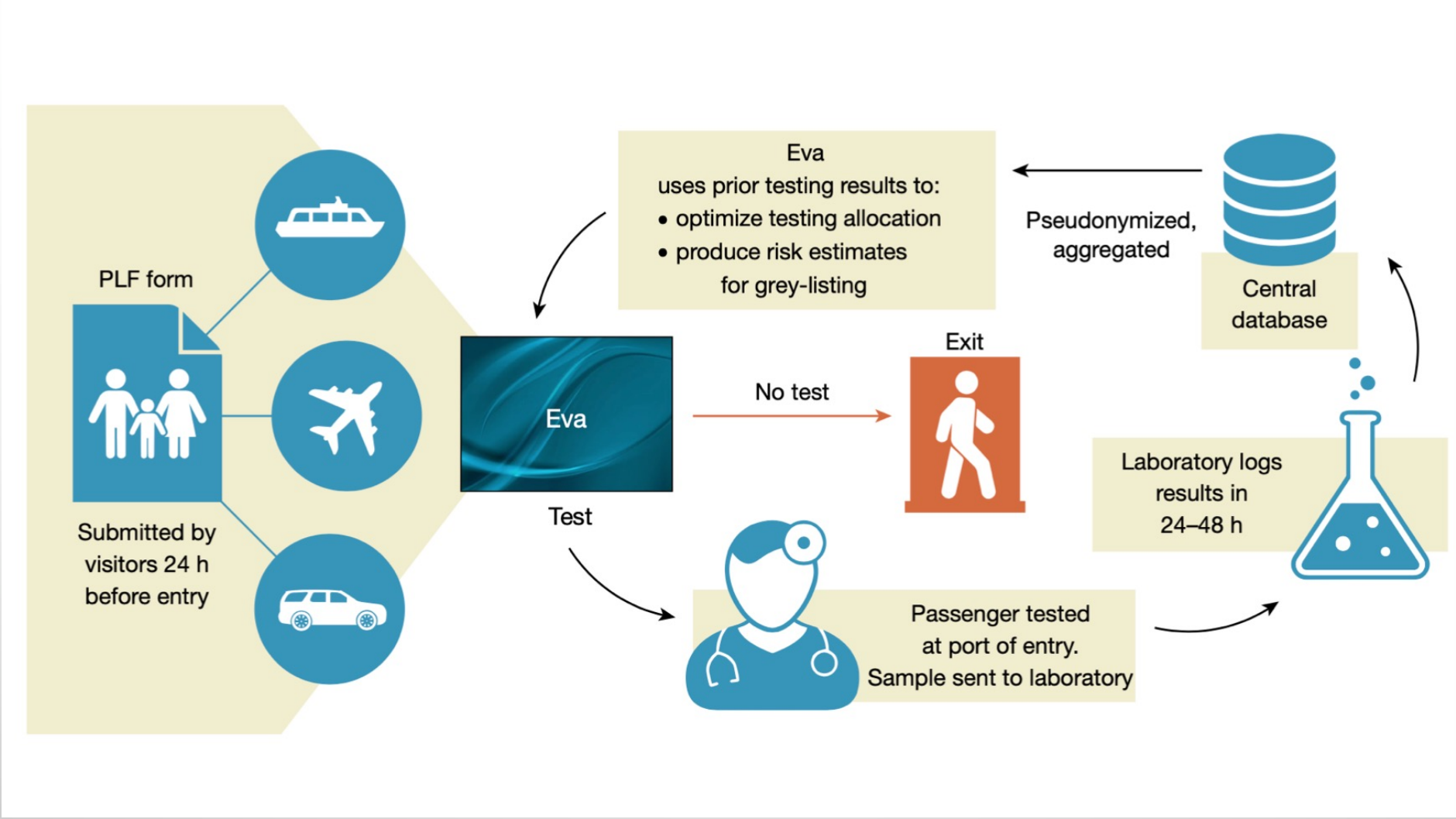
Prereqs.

- Stochastic Processes (Prob. And Stats, Markov Processes, Estimation Theory, Information Theory)
- Optimization (Lagrange Multipliers)
- Deep Learning (Concepts and Pytorch)

Motivation (cont.) ChatGPT; Why RL?!



Motivation (cont.)

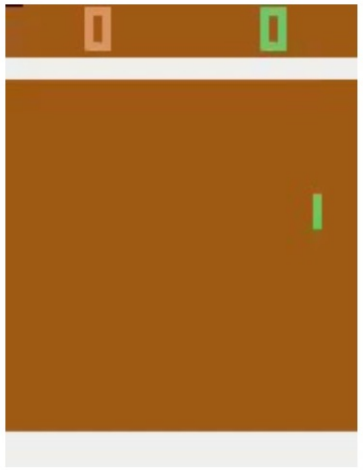


- https://www.nature.com/articles/s41586-021-04014-z

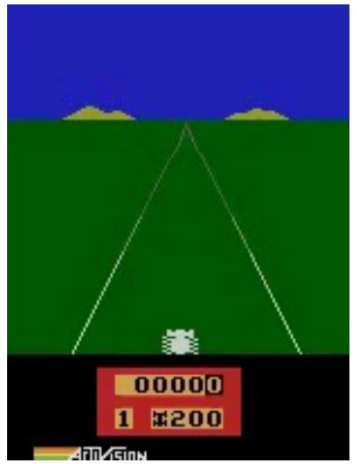
History

2013

Atari (DQN)
[Deepmind]

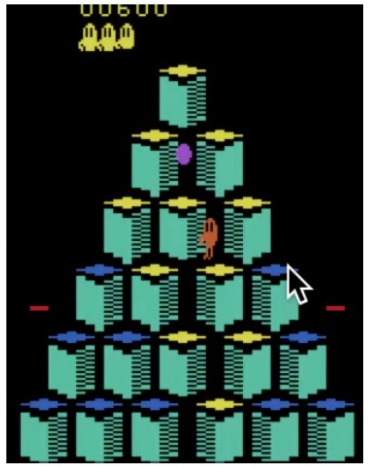


Pong



Enduro

Beamrider

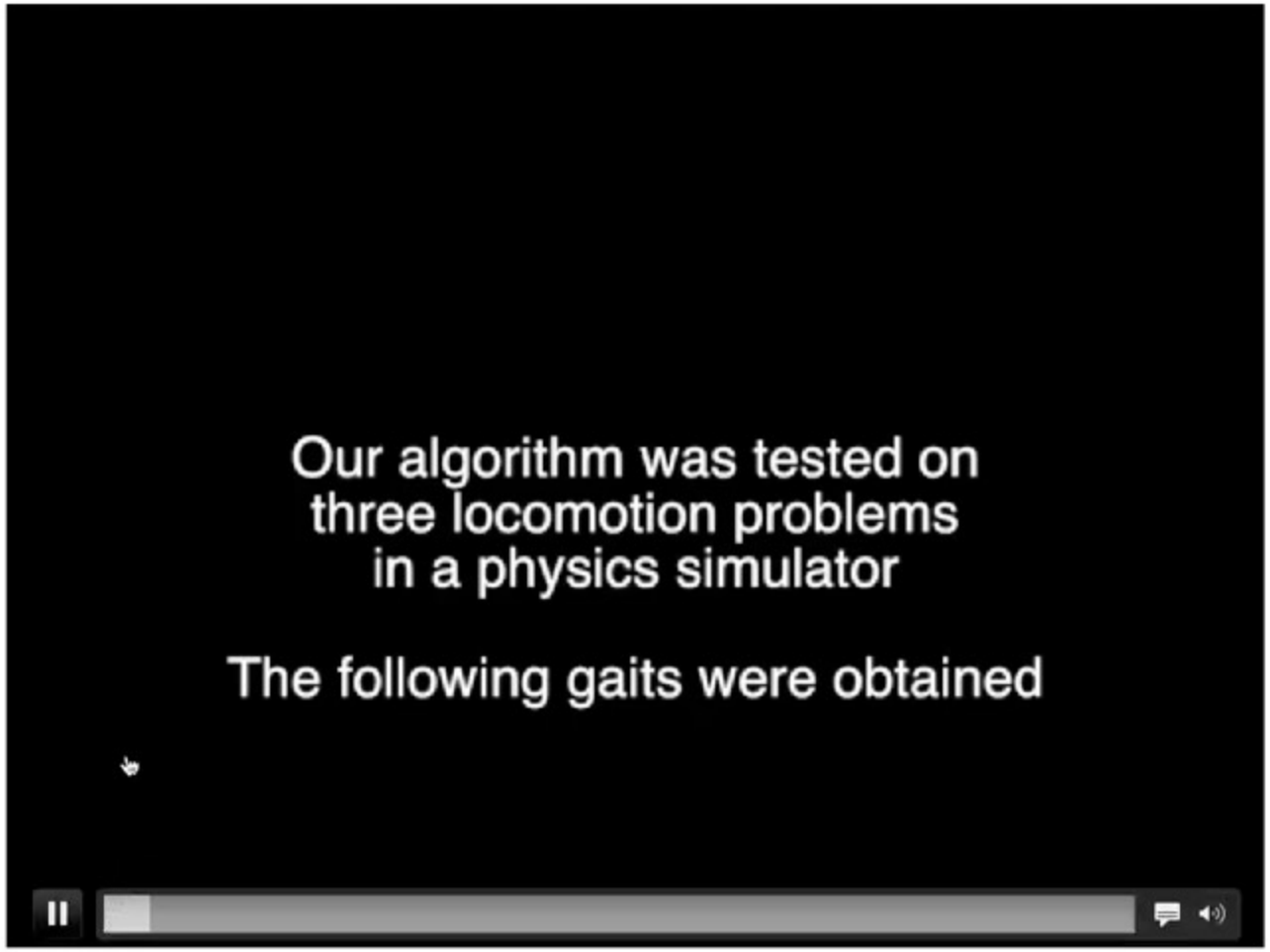


Q*bert

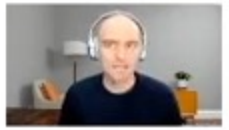
A Few Deep RL Highlights

2013 Atari (DQN) [Deepmind]

2014 2D locomotion (TRPO) [Berkeley]



Play 0:06 – 0:25



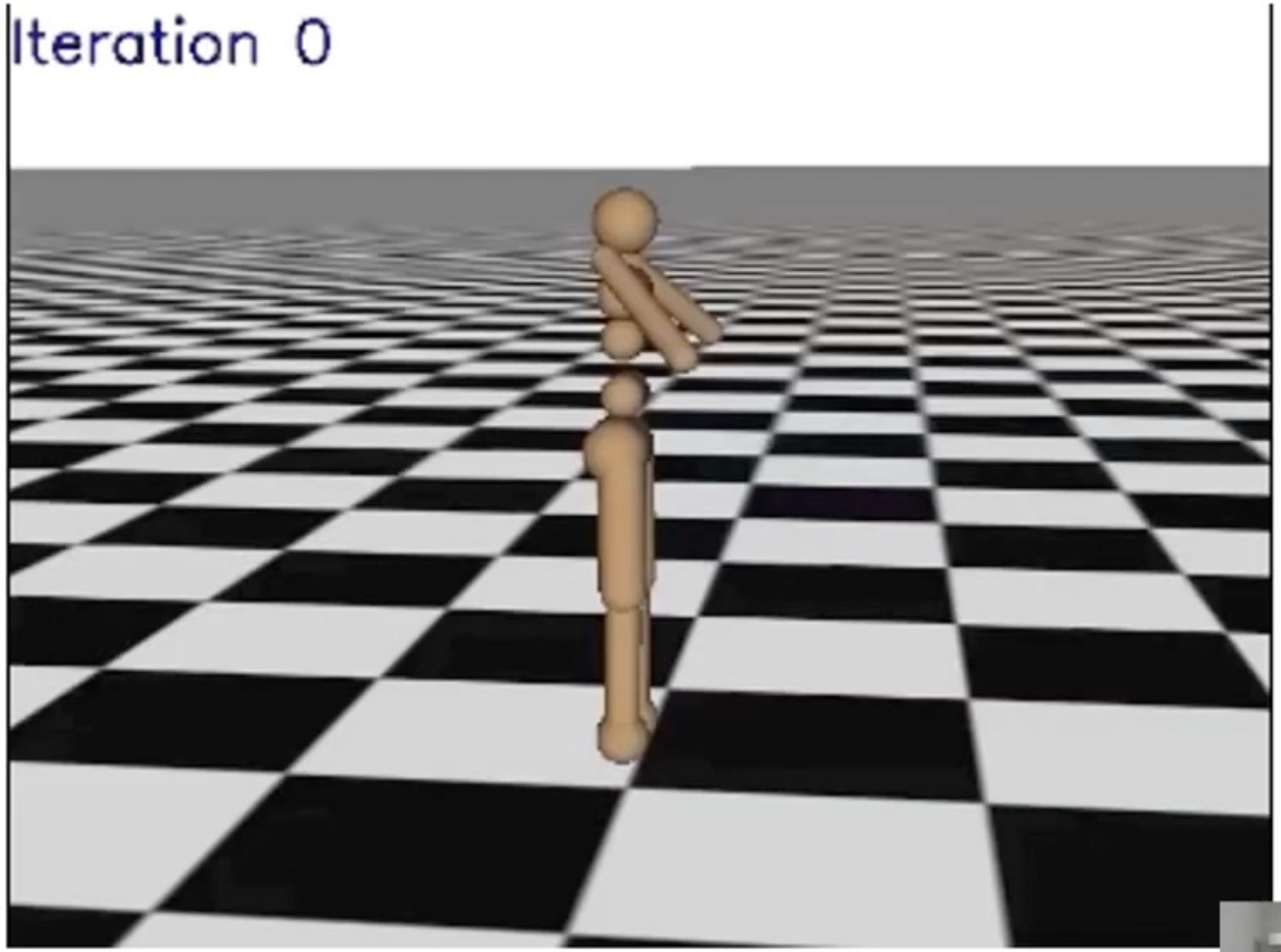
History

2013	Atari (DQN) [Deepmind]
2014	2D locomotion (TRPO) [Berkeley]
2015	**AlphaGo [Deepmind]**

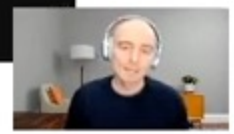
Tian et al, 2016; Maddison et al, 2014; Clark et al, 2015

A Few Deep RL Highlights

2013	Atari (DQN) [Deepmind]
2014	2D locomotion (TRPO) [Berkeley]
2015	AlphaGo [Deepmind]
2016	**3D locomotion (TRPO+GAE) [Berkeley]**



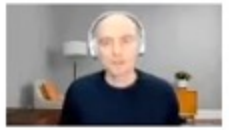
[Schulman, Moritz, Levine, Jordan, Abbeel, ICLR 2016]



A Few Deep RL Highlights

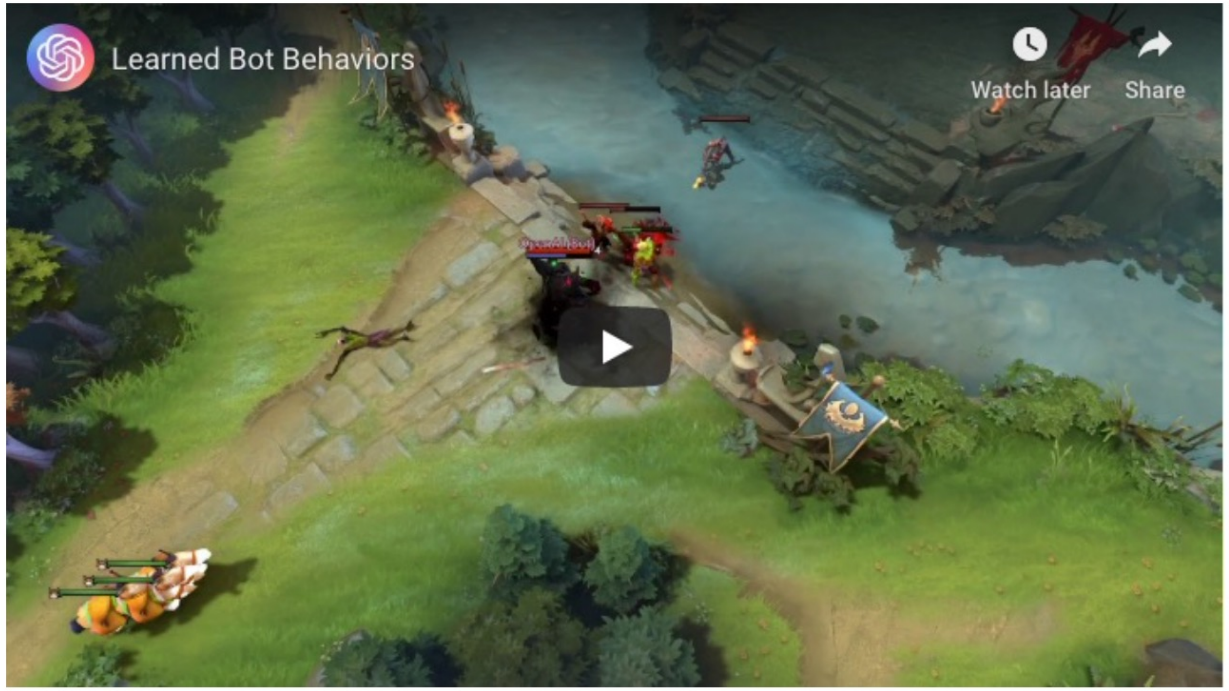
2013	Atari (DQN) [Deepmind]
2014	2D locomotion (TRPO) [Berkeley]
2015	AlphaGo [Deepmind]
2016	3D locomotion (TRPO+GAE) [Berkeley]
2016	**Real Robot Manipulation (GPS) [Berkeley]**

[Levine*, Finn*, Darrell, Abbeel, JMLR 2016]



History

2013	Atari (DQN) [Deepmind]
2014	2D locomotion (TRPO) [Berkeley]
2015	AlphaGo [Deepmind]
2016	3D locomotion (TRPO+GAE) [Berkeley]
2016	Real Robot Manipulation (GPS) [Berkeley, Google]
2017	**Dota2 (PPO) [OpenAI]**



OpenAI Dota Bot beat best humans 1:1 (Aug 2018)

A Few Deep RL Highlights

2013 Atari (DQN) [Deepmind]

2014 2D locomotion (TRPO) [Berkeley]

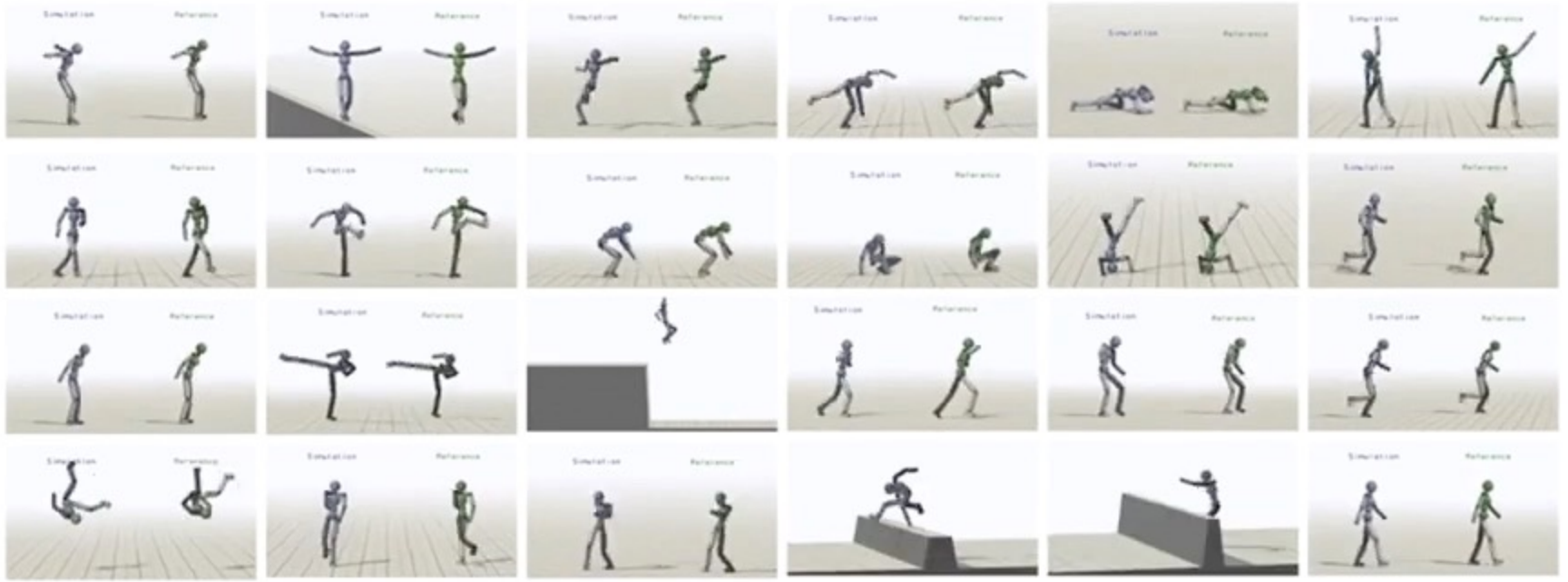
2015 AlphaGo [Deepmind]

2016 3D locomotion (TRPO+GAE) [Berkeley]

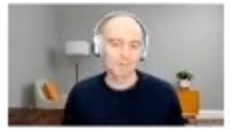
2016 Real Robot Manipulation (GPS) [Berkeley, Google]

2017 Dota2 (PPO) [OpenAI]

2018 DeepMimic [Berkeley]



[Peng, Abbeel, Levine, van de Panne, 2018]

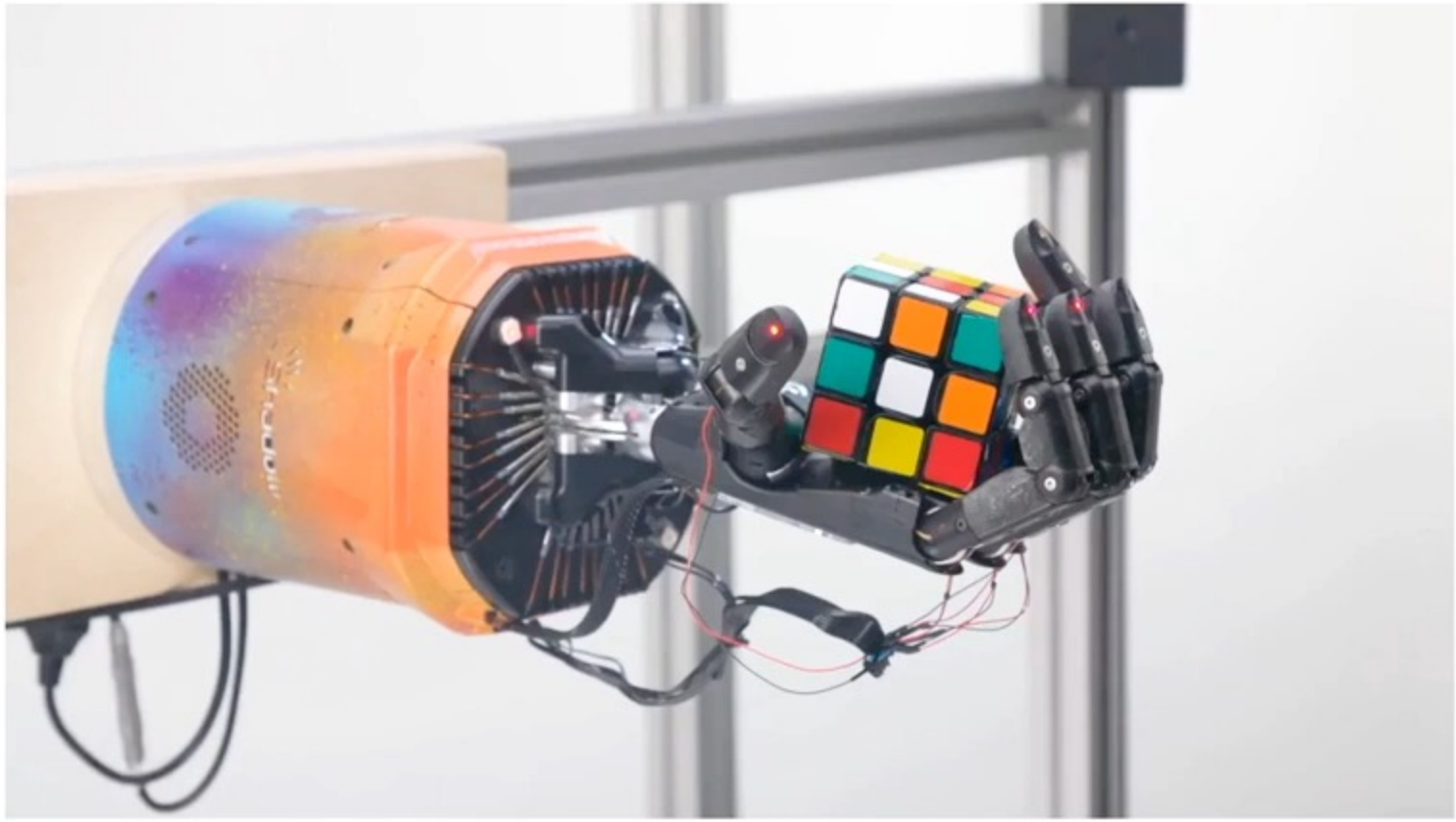


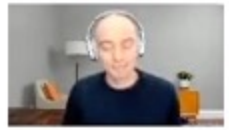
History

2013	Atari (DQN) [Deepmind]
2014	2D locomotion (TRPO) [Berkeley]
2015	AlphaGo [Deepmind]
2016	3D locomotion (TRPO+GAE) [Berkeley]
2016	Real Robot Manipulation (GPS) [Berkeley, Google]
2017	Dota2 (PPO) [OpenAI]
2018	DeepMimic [Berkeley]
2019	**AlphaStar [Deepmind]**

A Few Deep RL Highlights

2013	Atari (DQN) [Deepmind]
2014	2D locomotion (TRPO) [Berkeley]
2015	AlphaGo [Deepmind]
2016	3D locomotion (TRPO+GAE) [Berkeley]
2016	Real Robot Manipulation (GPS) [Berkeley, Google]
2017	Dota2 (PPO) [OpenAI]
2018	DeepMimic [Berkeley]
2019	AlphaStar [Deepmind]
2019	**Rubik's Cube (PPO+DR) [OpenAI]**

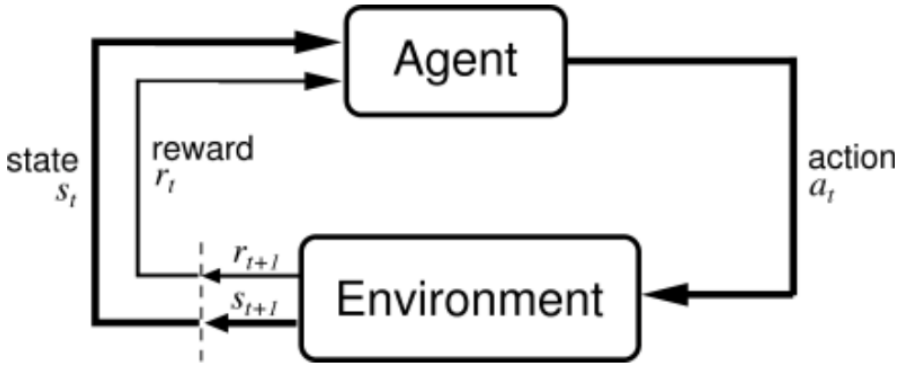




Let's Begin: Markov Decision Processes (MDPs)

An MDP is defined by:

- Set of states S
- Set of actions A
- Transition function $P(s' \mid s, a)$
- Reward function $R(s, a, s')$
- Start state s_0
- Discount factor γ
- Horizon H



The Goal

- The policy is $\pi_\theta : S \to A$ for infinite horizon or

  $\pi_\theta : S \times \{0, \dots, H\} \to A$ for finite horizon MDP.

MDP (S, A, T, R, γ, H), goal: $max_\pi \mathrm{E}[\sum_{t=0}^{H} \gamma^t R(S_t, A_t, S_{t+1}) | \pi]$

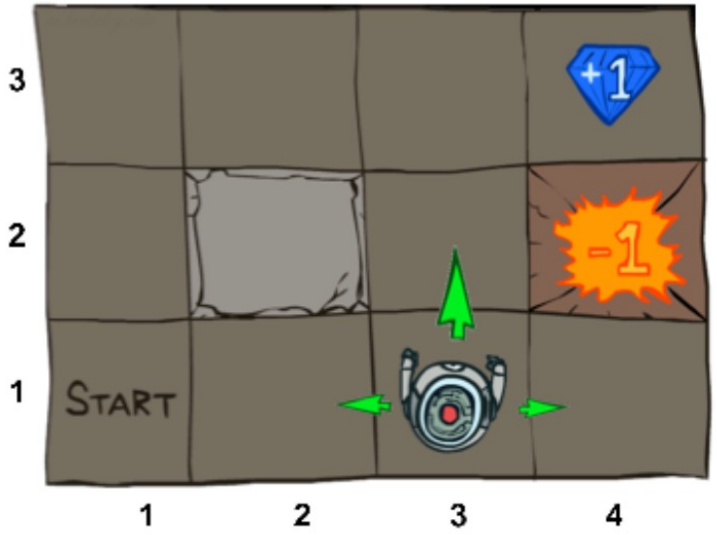
Sometimes the policy could be stochastic: $\pi : S \times A \to [0,1]$, which is

$\pi(a|s) = \Pr(A_t = a | S_t = s)$.

Example: Grid World

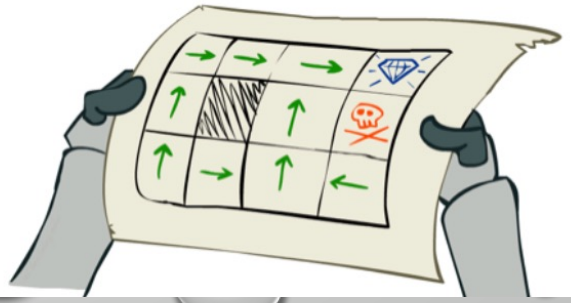
An MDP is defined by:

- Set of states S
- Set of actions A
- Transition function $P(s' \mid s, a)$
- Reward function $R(s, a, s')$
- Start state s_0
- Discount factor γ
- Horizon H

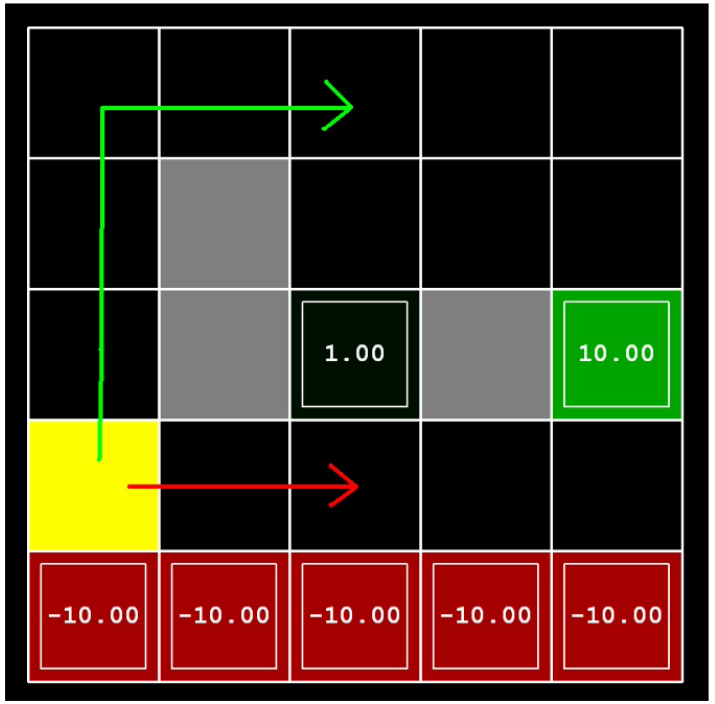


Goal: $$max_{\pi} \mathrm{E}[\sum_{t=0}^{H} \gamma^t R(S_t, A_t, S_{t+1}) | \pi]$$

π:



Exercise



(a) Prefer the close exit (+1), risking the cliff (-10)

(b) Prefer the close exit (+1), but avoiding the cliff (-10)

(c) Prefer the distant exit (+10), risking the cliff (-10)

(d) Prefer the distant exit (+10), avoiding the cliff (-10)

(1) $\gamma = 0.1$, noise = 0.5

(2) $\gamma = 0.99$, noise = 0

(3) $\gamma = 0.99$, noise = 0.5

(4) $\gamma = 0.1$, noise = 0