



# Introduction To Reinforcement Learning

Thien-Minh Nguyen, PhD

Centre for Advance Robotics Technology Innovation

Feb 2025

# Outline

- Recent RL applications in robotics.
- Overview of RL
- Basic Methods to solve the RL problems
- Tutorials
  - Tic-tac-toe. Complete MDP with *Value Iteration Technique*.
  - Cartpole. Small scaled complete RL problem → benchmarking and analysis

# Objectives

- Exposure to mathematical formulism of RL.
- Familiarize with basic concepts of Reinforcement Learning (RL).

In the context of RL...

- Agent, environment, observations, state, reward, action, value, return, discount ...
- Evaluation, Iteration, Improvement, Value Iteration ...
- Monte Carlo, Off-policy
- Temporal Difference, Q-learning, Sarsa
- Function Approximation
- Policy Gradient Methods

# Deep Reinforcement Learning Doesn't Work Yet

Feb 14, 2018

June 24, 2018 note: If you want to cite an example from the post, please cite the paper which that example came from. If you want to cite the post as a whole, you can use the following BibTeX:

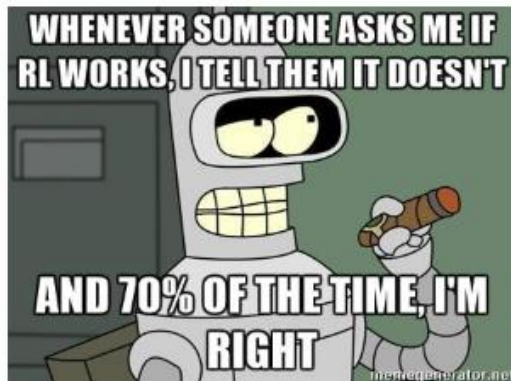
```
@misc{rlblogpost,  
  title={Deep Reinforcement Learning Doesn't Work Yet},  
  author={Irpan, Alex},  
  howpublished={\url{https://www.alexirpan.com/2018/02/14/rl-hard.html}},  
  year={2018}  
}
```

This mostly cites papers from Berkeley, Google Brain, DeepMind, and OpenAI from the past few years, because that work is most visible to me. I'm almost certainly missing stuff from older literature and other institutions, and for that I apologize - I'm just one guy, after all.

## Introduction

Once, on Facebook, I made the following claim.

Whenever someone asks me if reinforcement learning can solve their problem, I tell them it can't. I think this is right at least 70% of the time.



<https://www.alexirpan.com/2018/02/14/rl-hard.html>



**Alexander Irpan** ✓ · 3rd

Research Scientist at Google

Google · University of California, Berkeley

Berkeley, California, United States · [Contact info](#)

266 connections

Message








+ Follow



Connect if you know each other

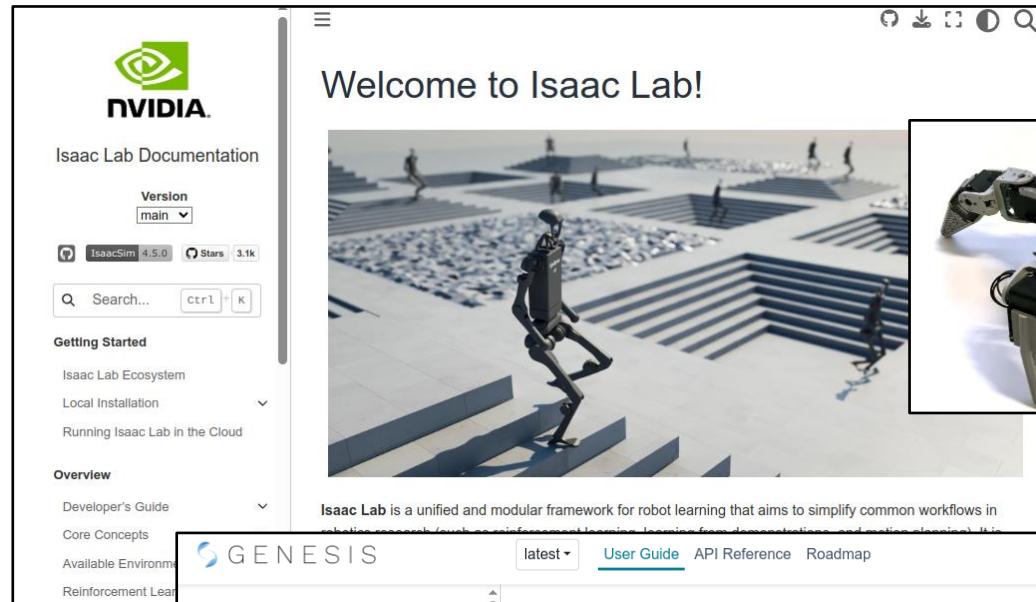
Connect

# Why RL ain't work?

-  Sample Inefficient
-  Can be solved by other methods
-  Always requires a reward function
-  Reward function design is difficult
-  Local optima hard to escape
-  Overfitting
-  Unstable and hard to reproduce

# Why RL works now?

- Sample Inefficient → **Cost of experiment ↓**




**\$75,000**  
Pricing considerations

The Spot robot dog price isn't budget-friendly: Spot **starts at \$75,000**. That's before adding payloads, sensors, or software packages. 3 Mar 2025

Standard Bots  
<https://standardbots.com> › blog › spot-robot

[The Spot Robot by Boston Dynamics: Features & Use Cases](#)



The ANYmal robot, a four-legged robot made by ANYbotics for industrial customers, costs **around \$150,000** and includes the full autonomy platform with LIDAR and a docking station, but excludes payloads, self-charging docks, and autonomous capabilities. [Link](#)



**GENESIS** latest User Guide API Reference Roadmap

Home > User Guide > Training Locomotion Policies with RL

## Training Locomotion Policies with RL

Genesis supports parallel simulation, making it ideal for training reinforcement learning (RL) locomotion policies efficiently. In this tutorial, we will walk you through a complete training pipeline to obtain a basic locomotion policy that enables a Unitree Go2 Robot to walk.

This is a simple and minimal example that demonstrates a very basic RL training pipeline in parallel simulation. With the following example you will be able to obtain a quadruped locomotion policy that is deployable to a real robot very quickly.

**Note:** This is *NOT* a comprehensive locomotion policy training pipeline. It uses simplified reinforcement learning terms to get you started easily, and does not exploit Genesis's speed on big batchsizes, so it serves basic demonstration purposes.

**Acknowledgement:** This tutorial is inspired by and builds several core concepts from [Legged Gym](#).

**Section Navigation**

**Overview**

- What is Genesis
- Why A New Physics Simulator
- Installation
- Genesis Vision & Mission

**Getting Started**

- Hello, Genesis
- Visualization & Rendering
- Control Your Robot
- Parallel Simulation
- Inverse Kinematics & Motion Planning
- Advanced and Parallel IK
- Beyond Rigid Bodies
- Interactive Information Access and

**UNITREE**

Home Professional Education & Industry Humanoid Robot Accessories Contact Disclaimer

Home > Unitree Go1

## Unitree Go1

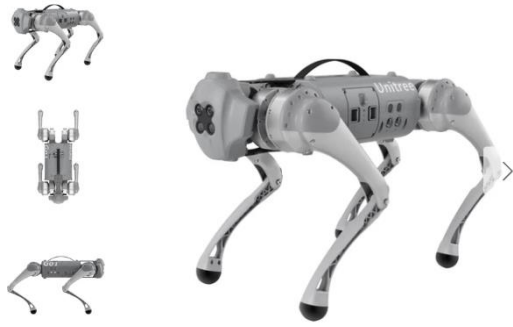
**\$2,700.00 USD**

- Ready to ship
- Shipping costs \$1000 per unit
- Duty not included, US needs to charge 25 percent duty
- Contact sales for Go1 Edu price

**Version**


Go1 Air Go1 pro

[Chat with us](#)



# Why RL *works* now?

- Sample Inefficient → **Cost of experiment** ↓
- Some problems can be solved by other methods  
→ **and many others can be solved by RL**




[Home](#) > [Nanyang Business School](#) > [News & Events](#) > [News](#)

Published on 14 Nov 2023

**NBS Knowledge Lab Interdisciplinary Distinguished Speaker Series webinar: Reinforcement Learning for Quantitative Trading**


## AlphaGo

AlphaGo mastered the ancient game of Go, defeated a Go world champion, and inspired a new era of AI systems.



[README](#) [MIT license](#)

[DeepSeek-R1](#)



**nature**

[Explore content](#) [About the journal](#) [Publish with us](#)

[nature](#) > [articles](#) > [article](#)

Article | [Open access](#) | Published: 05 October 2022

**Discovering faster matrix multiplication algorithms with reinforcement learning**

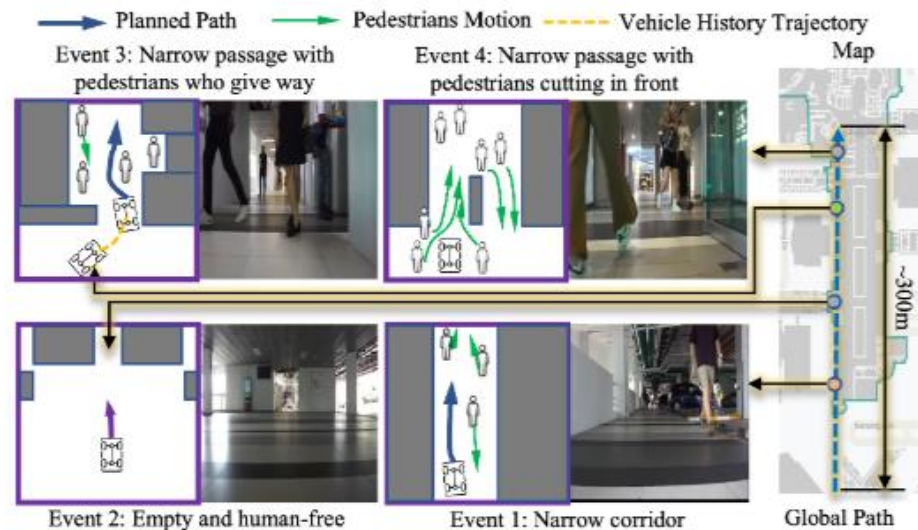


# Why RL works now?

- Sample Inefficient → **Cost of experiment** ↓
- Some problems can be solved by other methods  
→ **and many others can be solved by RL**

## Learning Dynamic Weight Adjustment for Spatial-Temporal Trajectory Planning in Crowd Navigation

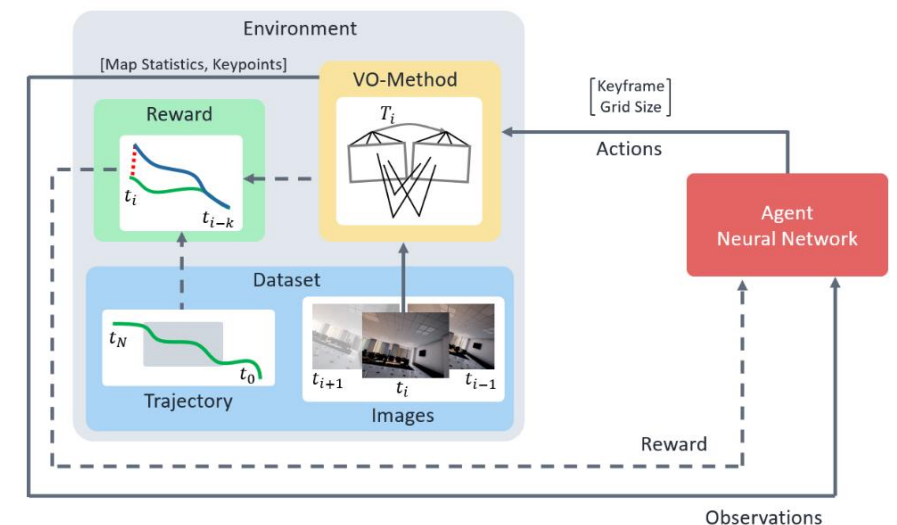
Muqing Cao\*, Xinhang Xu\*, Yizhuo Yang\*, Jianping Li, Tongxing Jin, Pengfei Wang,  
Tzu-Yi Hung, Guosheng Lin, and Lihua Xie<sup>1</sup> *Fellow, IEEE*



## Reinforcement Learning Meets Visual Odometry

Nico Messikommer\*, Giovanni Cioffi\*,  
Mathias Gehrig, and Davide Scaramuzza

Dept. of Informatics, University of Zurich  
 {nmessi,cioffi,mgehrig,sdavid}@ifi.uzh.ch





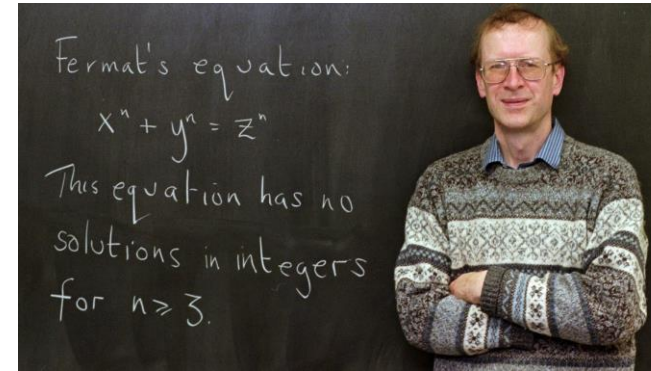
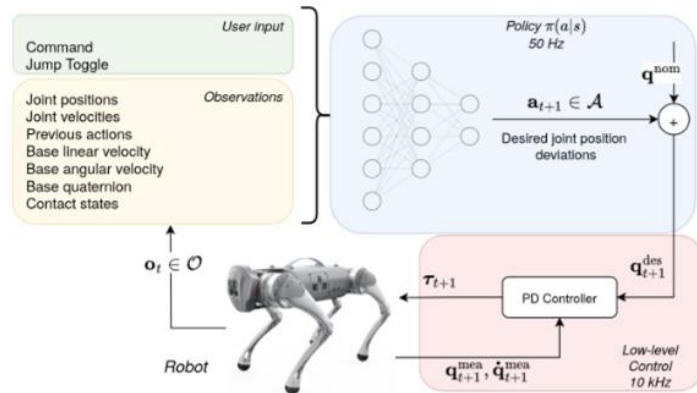
# Why RL *works* now?

- Sample Inefficient → **Cost of experiment** ↓
- Some problems can be solved by other methods  
→ **and many others can be solved by RL**



# Why RL works now?

- Sample Inefficient → **Cost of experiment** ↓
- Some problems can be solved by other methods → **Some can be solved by RL**
- Always requires a reward function
- Reward function design is difficult
- Local optima hard to escape
- Overfitting
- Unstable and hard to reproduce



## Curriculum-Based Reinforcement Learning for Quadrupedal Jumping: A Reference-free Design

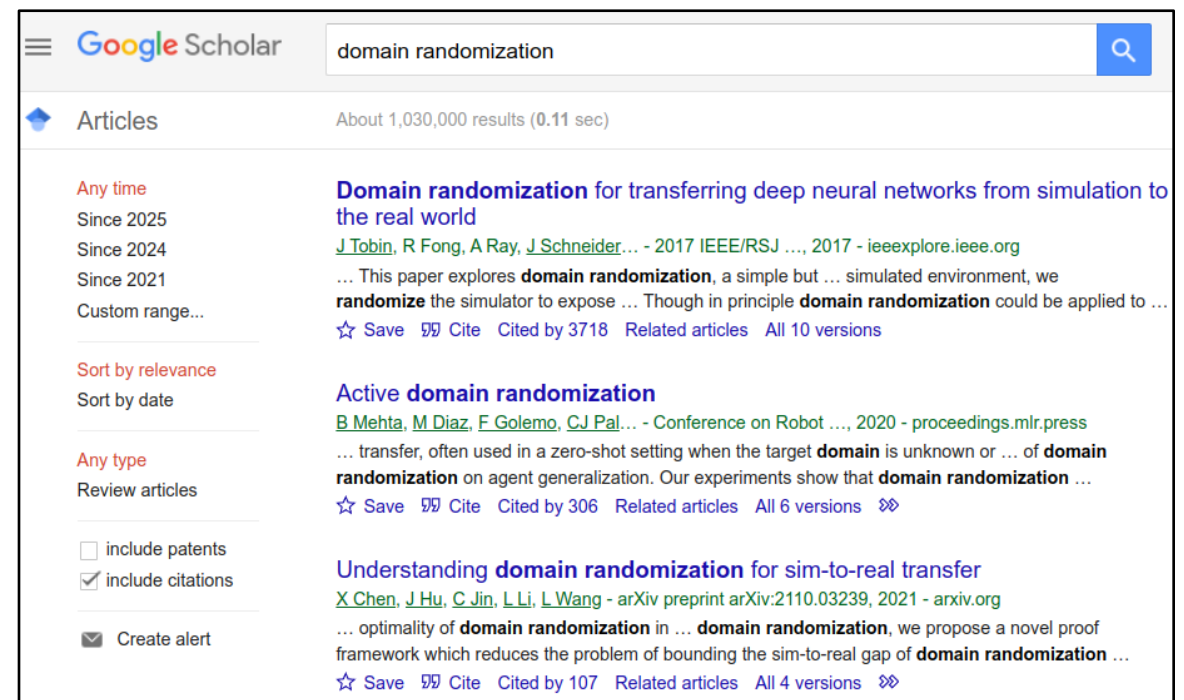
Vassil Atanassov\*, Jiatao Ding\*, Jens Kober, Ioannis Havoutis, Cosimo Della Santina

REWARDS DEFINITION. THE LIGHT ORANGE COLOUR INDICATES TASK-BASED REWARDS, WHILE THE LIGHT PURPLE SHADE DESCRIBES REGULARISATION REWARDS.  $w_x$  IS THE WEIGHT,  $\sigma_x$  IS A SCALING FACTOR FOR THE EXPONENTIAL KERNEL,  $e(\cdot)$  AND  $\log(\cdot)$  SEPARATELY DENOTE THE EXPONENT AND LOGARITHM OPERATION.

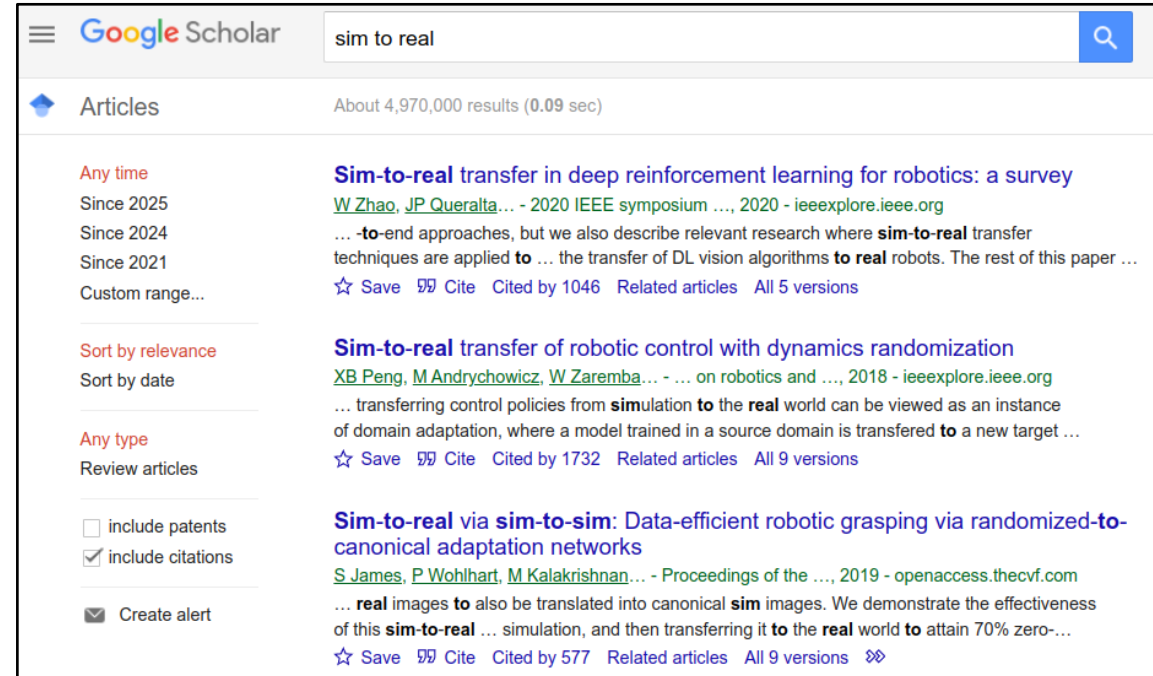
Name	Type	Stance	Flight	Landing
Landing position	Single	0	0	$w_p(e(-\sum   p_{land} - p_{des}  ^2)/\sigma_{p,land})$
Landing orientation	Single	0	0	$w_{ori}(e(-  \log(\bar{q}_{land}^{-1} * \bar{q}_{des})  ^2)/\sigma_{ori,land})$
Max height	Single	0	0	$w_h(e(  h_{max} - 0.9  ^2)/\sigma_{p_z,max})$
Jumping	Single	0	0	$w_{jump}$
Base Position	Continuous	$w_{p_z,st}(e(-  p_z - 0.20  ^2/\sigma_{p_z,st}))$	$w_{p_z,f}(e(-  p_z - 0.7  ^2/\sigma_{p_z,f}))$	$w_{p,l}(e(-\sum   p - p_{des}  ^2/\sigma_{p,l}))$
Orientation Tracking	Continuous	$w_{ori,st}(e(-  \log(\bar{q}_{base}^{-1} * \bar{q}_{des})  ^2/\sigma_{ori,st}))$	0	$w_{ori,l}(e(-  \log(\bar{q}_{base}^{-1} * \bar{q}_{des})  ^2/\sigma_{ori,l}))$
Base linear velocity	Continuous	0	$w_{v_x,y}(e(-\sum   v_{x,y} - v_{des}  ^2/\sigma_v))$	0
Base angular velocity	Continuous	0	$w_{\omega}(e(-\sum   \omega - \omega_{des}  ^2/\sigma_{\omega}))$	$0.1w_{\omega}(e(-\sum   \omega  ^2/\sigma_{\omega}))$
Feet clearance	Continuous	0	$w_{feet}( p_{feet} - p_{feet}^0  + [0.0, 0.0, -0.15])^2$	0
Symmetry	Continuous	$w_{sym}(\sum_{joint}   q_{left} - q_{right}  ^2)$		
Nominal pose	Continuous	$w_q(e(-\sum_{joint}   q_j - q_{j,nom}  ^2/\sigma_q))$	$0.1w_q(e(-\sum_{joint}   q_j - q_{j,nom}  ^2/\sigma_q))$	$w_q(e(-\sum_{joint}   q_j - q_{j,nom}  ^2/\sigma_q))$
Energy	Continuous	$w_{energy}(\tau^T \dot{q})$		
Base acceleration	Continuous	$w_{acc} \dot{v} ^2$		
Contact change	Continuous	$w_c \sum_{feet} (c_{foot}(t) - c_{foot}(t-1))$		
Maintain Contact	Continuous	$w_{contact} \sum_{feet} c_{foot}(t)$	0	0
Contact forces	Continuous	$w_{F_c} \sum_{i=0}^{n_f}  F_i - \bar{F} $		
Action rate	Continuous	$w_a \sum_{joint}  a(t) - a(t-1) ^2$		
Joint acceleration	Continuous	$w_{\ddot{q}} \sum_{joint}  \ddot{q}_j ^2$		
Joint limits	Continuous	$w_{qlim} \sum_{joint}  q_j - q_{j,lim} ^2$		

# Why RL *works* now?

- Sample Inefficient → **Cost of experiment** ↓
  - Some problems can be solved by other methods → **Some can be solved by RL**
  - Always requires a reward function
  - Reward function design is difficult
  - Local optima hard to escape
  - Overfitting
  - Unstable and hard to reproduce
- **Active research areas**



Google Scholar search results for "domain randomization". The search bar shows "domain randomization" and the results are sorted by relevance. The first article is "Domain randomization for transferring deep neural networks from simulation to the real world" by J Tobin, R Fong, A Ray, J Schneider, et al. (2017 IEEE/RSJ ...). The second article is "Active domain randomization" by B Mehta, M Diaz, F Golemo, C J Pal, et al. (2020 - proceedings.mlr.press). The third article is "Understanding domain randomization for sim-to-real transfer" by X Chen, J Hu, C Jin, L Li, L Wang (2021 - arxiv.org).



Google Scholar search results for "sim to real". The search bar shows "sim to real" and the results are sorted by relevance. The first article is "Sim-to-real transfer in deep reinforcement learning for robotics: a survey" by W Zhao, JP Queralta, et al. (2020 IEEE symposium ...). The second article is "Sim-to-real transfer of robotic control with dynamics randomization" by XB Peng, M Andrychowicz, W Zaremba, et al. (2018 - ieexplore.ieee.org). The third article is "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks" by S James, P Wohlhart, M Kalakrishnan, et al. (2019 - openaccess.thecvf.com).

# Why RL *works* now?

- Sample Inefficient → **Cost of experiment** ↓
  - Some problems can be solved by other methods → **Some can be solved by RL**
  - Always requires a reward function
  - Reward function design is difficult
  - Local optima hard to escape
  - Overfitting
  - Unstable and hard to reproduce
- **Active research areas**



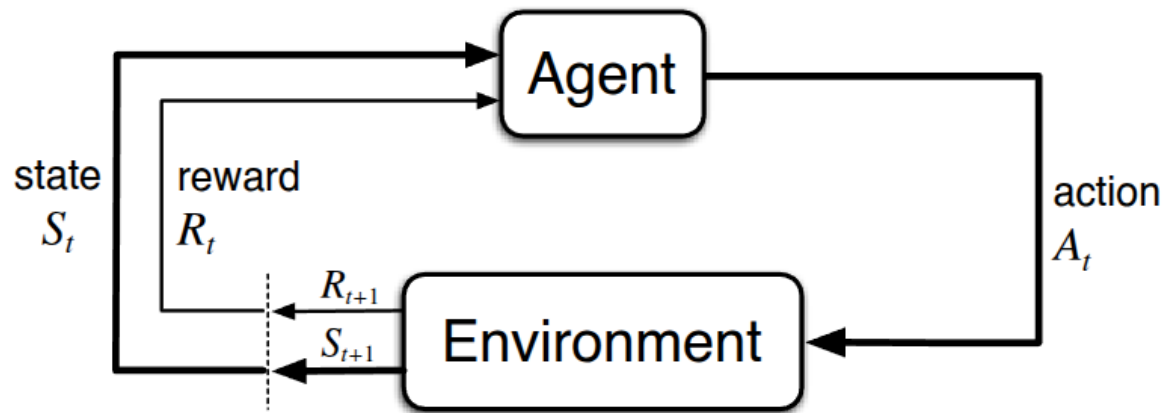
In the context of RL...

- Agent, environment, observations, state, reward, action, value, return, discount ...
- Evaluation, Iteration, Improvement, Value Iteration ...
- Monte Carlo, Off-policy
- Temporal Difference, Q-learning, Sarsa
- Function Approximation
- Policy Gradient Methods



# Agent and Environment

- **Agent:** receives **observations** and **rewards**, generates **action**.
- **Environment:** receives **action**, produces **observation** and **reward**.



The robot belongs to which category?

*"All goals can be described by the maximization of expected cumulative reward"*

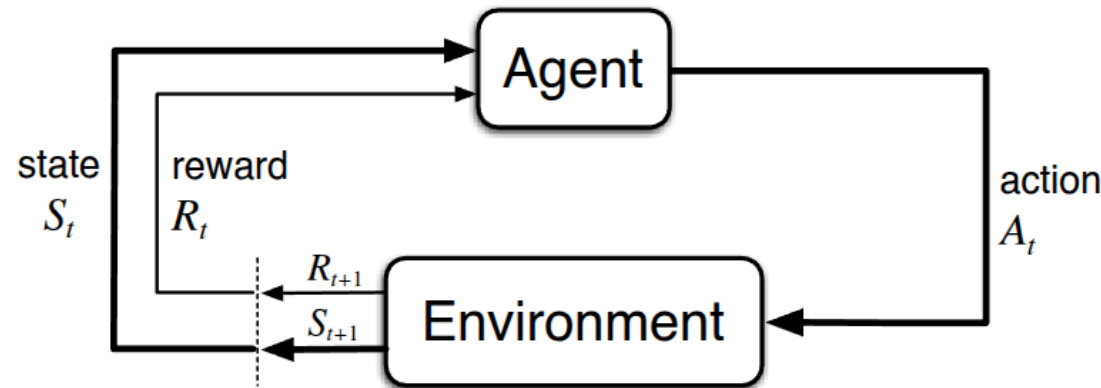
main IsaacLab / source / isaacsim\_tasks / isaacsim\_tasks / direct / anymal\_c / anymal\_c\_env\_cfg.py

Code Blame 148 lines (130 loc) · 4.48 KB · 1

```
53 class AnymalCFlatEnvCfg(DirectRLEnvCfg):
54     decimation = 4
55     action_scale = 0.5
56     action_space = 12
57     observation_space = 48
58     state_space = 0
59
60     # simulation
61
62     > sim: SimulationCfg = SimulationCfg( ...
73     )
74     > terrain = TerrainImporterCfg( ...
86     )
87
88     # scene
89     scene: InteractiveSceneCfg = InteractiveSceneCfg(num_envs=4096, env_spacing=4.0, replicate_physics=True)
90
91     # events
92     events: EventCfg = EventCfg()
93
94     # robot
95     robot: ArticulationCfg = ANYMAL_C_CFG.replace(prim_path="/World/envs/env_*/Robot")
96     contact_sensor: ContactSensorCfg = ContactSensorCfg(
97         prim_path="/World/envs/env_*/Robot/.*", history_length=3, update_period=0.005, track_air_time=True
98     )
99
100     # reward scales
101     lin_vel_reward_scale = 1.0
102     yaw_rate_reward_scale = 0.5
103     z_vel_reward_scale = -2.0
104     ang_vel_reward_scale = -0.05
105     joint_torque_reward_scale = -2.5e-5
106     joint_accel_reward_scale = -2.5e-7
107     action_rate_reward_scale = -0.01
108     feet_air_time_reward_scale = 0.5
109     undesired_contact_reward_scale = -1.0
110     flat_orientation_reward_scale = -5.0
```



# Concepts (1)



## Definition

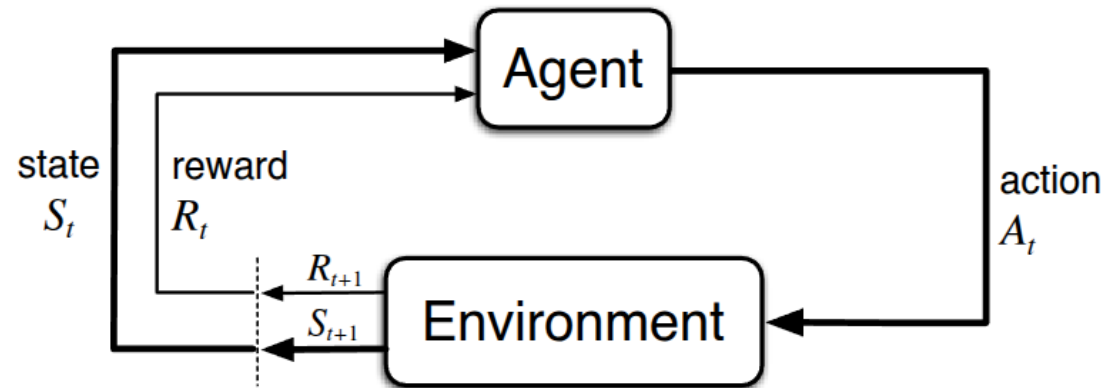
- A finite Markov Decision Process:
  - $R_t$  is the **reward**, a scalar signal
  - $A_t$  is the **action**, e.g., torque command, velocity command, chess moves ...
  - $S_t$  is the **state**.
  - $t \in \{0, 1, 2 \dots\}$ ,  $s \in \mathcal{S}$ ,  $a \in \mathcal{A}(s)$ ,  $r \in \mathcal{R} \subset \mathbb{R}$

- The dynamics between agent and environment is specified as:

$$\mathcal{P} \triangleq p(s', r | s, a) = \text{Prob}(S_{t+1} = s', R_{t+1} | S_t = s, A_t = a)$$

- $H_t \triangleq (S_1, R_1, A_1 \dots S_t, R_t, A_t)$ , the **trajectory**.
- $O_t = h(H_t)$  is the **observation**, e.g., image, can be IMU reading, lidar scan ...

# Concepts (1)



## Definition

- A finite Markov Decision Process:
  - $R_t$  is the **reward**, a scalar signal
  - $A_t$  is the **action**, e.g., torque command, velocity command, chess moves ...
  - $S_t$  is the **state**.
  - $t \in \{0, 1, 2 \dots\}$ ,  $s \in \mathcal{S}$ ,  $a \in \mathcal{A}(s)$ ,  $r \in \mathcal{R} \subset \mathbb{R}$
  - The dynamics between agent and environment is summarized in:

$$\mathcal{P} \triangleq p(s', r | s, a) = \text{Prob}(S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a)$$

- $H_t \triangleq (S_0, R_0, A_0 \dots S_t, R_t, A_t)$ , the **trajectory**.
- $O_t = h(H_t)$  is the **observation**, e.g., image, can be IMU reading, lidar scan ...

# Concepts (2) <sup>[1]</sup>

## Definition

- Policy function  $\pi(\cdot)$ :

- Deterministic policy:

$$a = \pi(s)$$

- Stochastic policy:

$$\pi(a|s) = \text{Prob}(A_t = a|S_t = s)$$

- The return  $G_t$ :

$$G_t \triangleq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$

- The discount factor:

$$\gamma \in [0, 1]$$

## Definition

- (State-)value function under policy  $\pi$ ,  $v_\pi(s)$ :

$$v_\pi(s) \triangleq E_\pi(G_t|S_t = s),$$

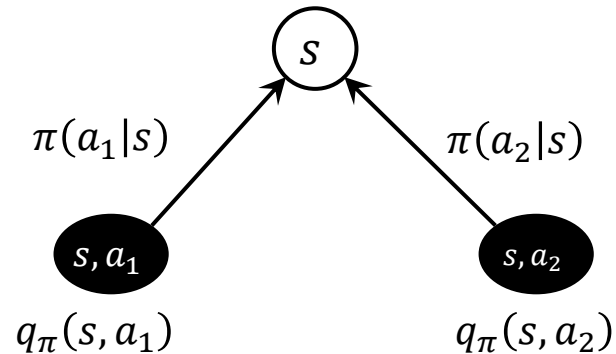
- The action-value function:

$$q_\pi(s, a) \triangleq E_\pi(G_t|S_t = s, A_t = a),$$

# Bellman Equation

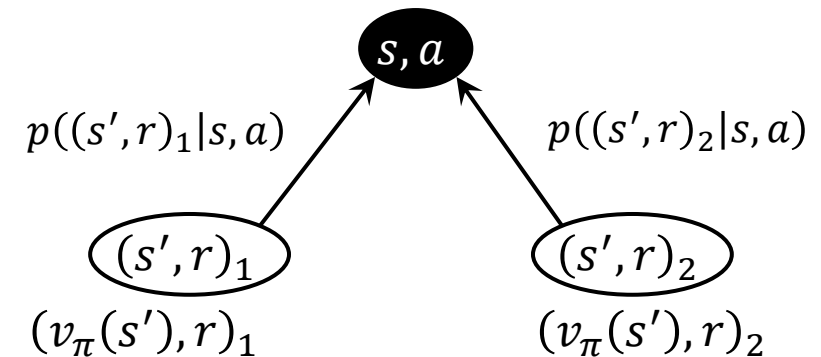
- For state-value function:

$$\begin{aligned} v_{\pi}(s) &= E_{\pi}(G_t | S_t = s) \\ &= \sum_{a \in \mathcal{A}} \pi(a|s) q_{\pi}(s, a) \end{aligned}$$



- For action-value function:

$$\begin{aligned} q_{\pi}(s, a) &= E_{\pi}(G_t | S_t = s, A_t = a) = E_{\pi}[R_{t+1} + \gamma v_{\pi}(S_{t+1}) | s, a] \\ &= \sum_{(s', r) \in \mathcal{S} \times \mathcal{R}} p(s', r | s, a) [r + \gamma v_{\pi}(s')] \end{aligned}$$



# Bellman Equation

- For state-value function:

$$\begin{aligned}
 v_{\pi}(s) &= E_{\pi}(G_t | S_t = s) \\
 &= \sum_{a \in \mathcal{A}} \pi(a|s) q_{\pi}(s, a) \\
 &= \sum_{a \in \mathcal{A}} \pi(a|s) \left[ \sum_{(s', r) \in \mathcal{S} \times \mathcal{R}} p(s', r | s, a) [r + \gamma v_{\pi}(s')] \right]
 \end{aligned}$$

- For action-value function:

$$\begin{aligned}
 q_{\pi}(s, a) &= E_{\pi}(G_t | S_t = s, A_t = a) = E_{\pi}[R_{t+1} + \gamma v_{\pi}(S_{t+1}) | s, a] \\
 &= \sum_{(s', r) \in \mathcal{S} \times \mathcal{R}} p(s', r | s, a) [r + \gamma v_{\pi}(s')] \\
 &= \sum_{(s', r) \in \mathcal{S} \times \mathcal{R}} p(s', r | s, a) \left[ r + \gamma \left[ \sum_{a' \in \mathcal{A}(s')} \pi(a'|s') q_{\pi}(s', a') \right] \right]
 \end{aligned}$$

# Optimal Value Functions & BOE

## Definition

- $\pi > \pi' \Rightarrow v_\pi(s) > v_{\pi'}(s), \forall s$
- The optimal state-value function  $v_*(s)$ :
- The optimal action-value function  $q_*(s, a)$ :
- For any optimal  $\pi_*$ , all  $s \in \mathcal{S}$ , all  $a \in \mathcal{A}(s)$ :

$$v_*(s) = \max_{\pi} v_\pi(s)$$

$$q_*(s) = \max_{\pi} q_\pi(s, a)$$

$$v_*(s) = \max_{a \in \mathcal{A}(s)} q_*(s, a)$$

$$q_*(s, a) = \sum_{(s', r) \in \mathcal{S} \times \mathcal{R}} p(s', r | s, a) [r + \gamma v_*(s)]$$

## Theorem

For any MDP:

- $\exists \pi_*, \pi_* \geq \pi, \forall \pi$
- $v_{\pi_*}(s) = v_*(s), \forall \pi_*$
- $q_{\pi_*}(s, a) = q_\pi(s, a), \forall \pi_*$

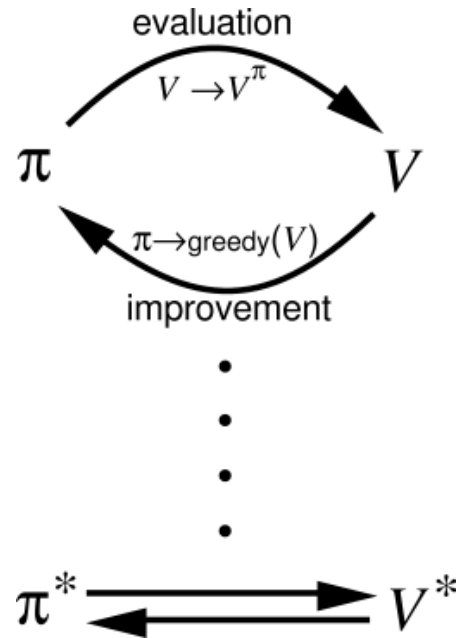
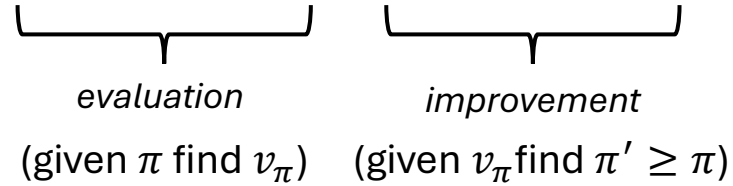
In the context of RL...

- Agent, environment, observations, state, reward, action, value, return, discount ...
- Evaluation, Iteration, Improvement, Value Iteration ...
- Monte Carlo, Off-policy
- Temporal Difference, Q-learning, Sarsa
- Function Approximation
- Policy Gradient Methods



# Solving the MDP

- Policy iteration: from some  $\pi \rightarrow$  evaluate  $\pi \rightarrow$  improve  $\pi$ , repeat until  $\pi \approx \pi^*$



- Value iteration: a direct approach that achieves faster convergence.

# Solving the MDP

## Policy *Evaluation*:

Given a policy  $\pi(a|s)$

- For  $k = 0 \dots K - 1$ :

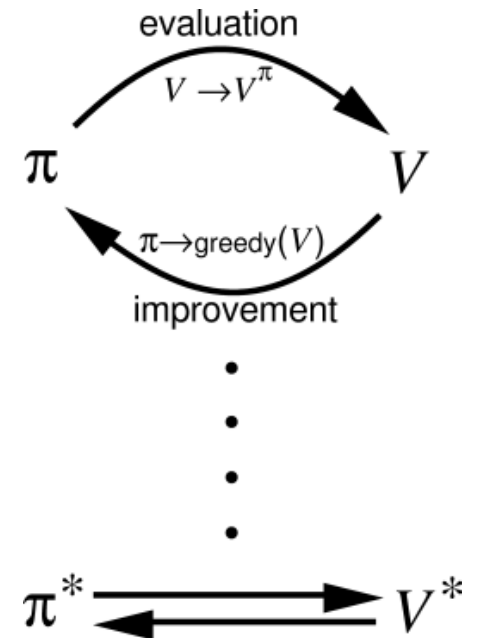
$$\forall s \in \mathcal{S}: V_{k+1}(s) \leftarrow \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{(s', r) \in \mathcal{S} \times \mathcal{R}} p(s', r|s, a) [r + \gamma V_k(s')],$$

- $V_k(s) \xrightarrow{K \rightarrow \infty} v_\pi(s)$

## Policy *Improvement*:

Given a value function  $v_\pi(s)$ :

- $\pi_* = \text{greedy}(v_\pi(s))$



**Policy Iteration**

# Solving the MDP

## Policy *Evaluation*:

Given a policy  $\pi(a|s)$

- For  $k = 0 \dots K - 1$ :

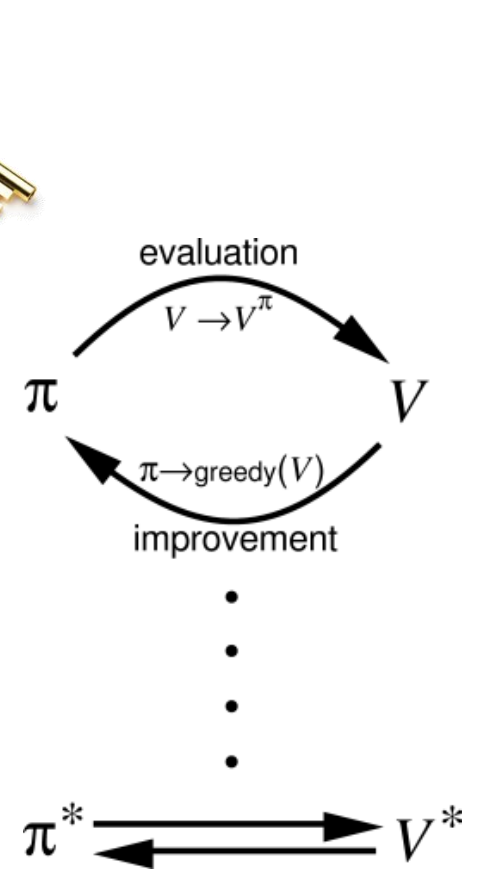
$$\forall s \in \mathcal{S}: V_{k+1}(s) \leftarrow \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{(s', r) \in \mathcal{S} \times \mathcal{R}} p(s', r|s, a) [r + \gamma V_k(s')],$$

- $V_k(s) \xrightarrow{K \rightarrow \infty} v_\pi(s)$

## Policy *Improvement*:

Given a value function  $v_\pi(s)$ :

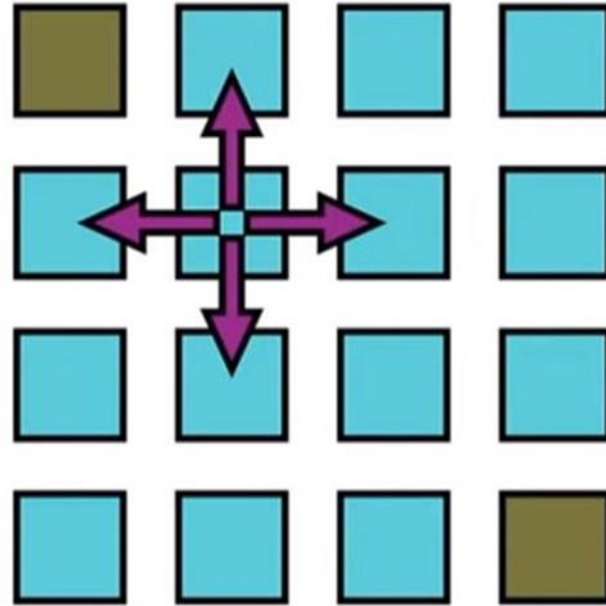
- $\pi_* = \text{greedy}(v_\pi(s))$



**Policy Iteration**

# Policy Iteration

Compute  $v_\pi(s)$  or  $q_\pi(s, a)$  for a given  $\pi$ .



$$R_t = -1$$

$$\pi(a|s) = 0.25$$

# Policy Iteration

Compute  $v_\pi(s)$  or  $q_\pi(s, a)$  for a given  $\pi$ .

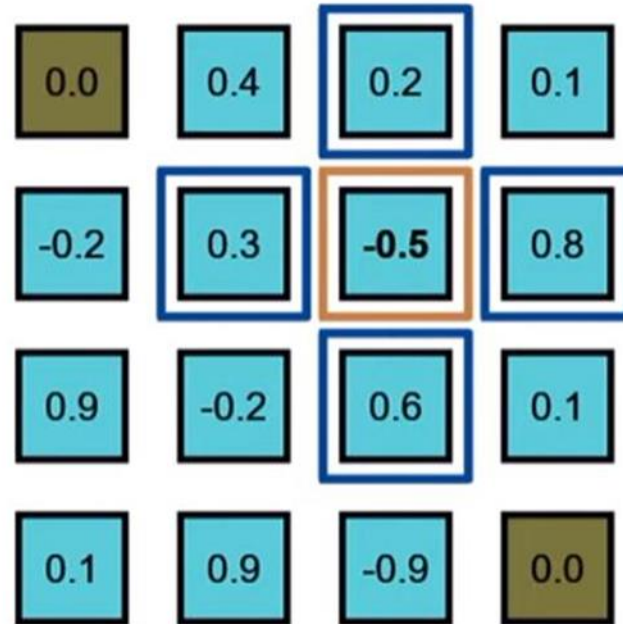
0.0	0.4	0.2	0.1
-0.2	0.3	-0.1	0.8
0.9	-0.2	0.6	0.1
0.1	0.9	-0.9	0.0

$$R_t = -1$$

$$\pi(a|s) = 0.25$$

# Policy Iteration

Compute  $v_\pi(s)$  or  $q_\pi(s, a)$  for a given  $\pi$ .



$$R_t = -1$$

$$\pi(a|s) = 0.25$$

$$V(s) \leftarrow \sum_{a \in \mathcal{A}(s)} \pi(a|s) \sum_{\substack{s' \in \mathcal{S} \\ r \in \mathcal{R}}} p(s', r|s, a) [r + \gamma V(s')]$$

# Policy Iteration

Compute  $v_\pi(s)$  or  $q_\pi(s, a)$  for a given  $\pi$ .

0.0	-0.8	0.2	0.1
-0.2	0.3	-0.1	0.8
0.9	-0.2	0.6	0.1
0.1	0.9	-0.9	0.0

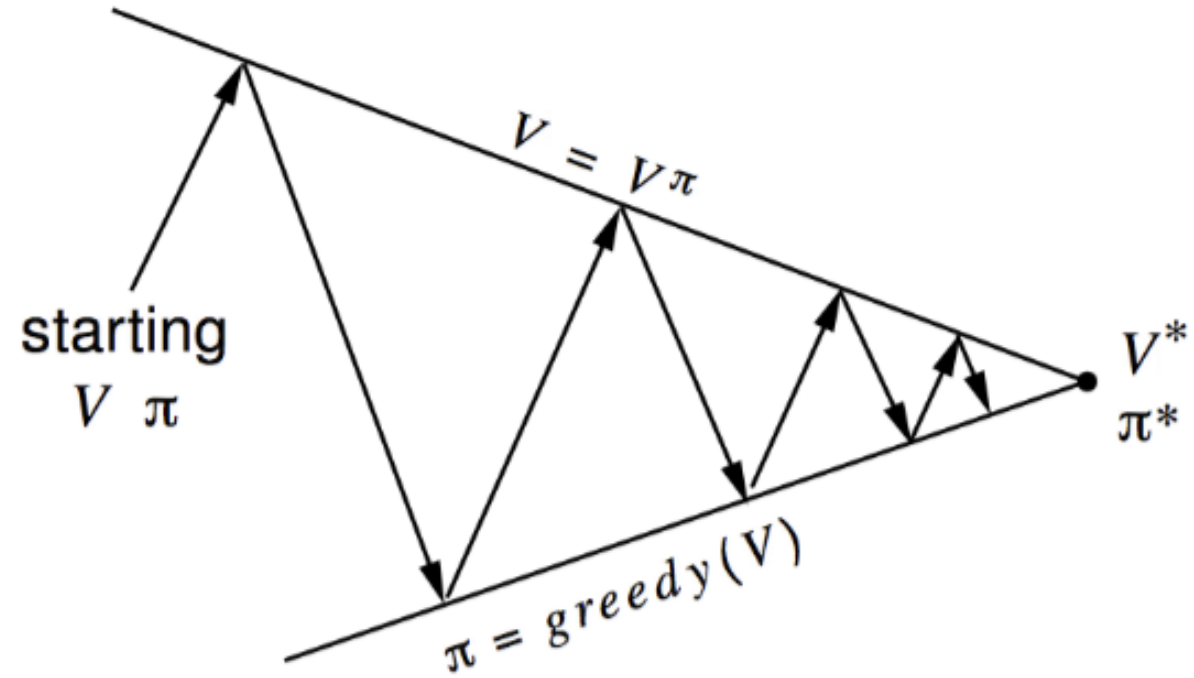
$$R_t = -1$$

$$\pi(a|s) = 0.25$$

$$V(s) \leftarrow \sum_{a \in \mathcal{A}(s)} \pi(a|s) \sum_{\substack{s' \in \mathcal{S} \\ r \in \mathcal{R}}} p(s', r|s, a) [r + \gamma V(s')]$$



# Policy Iteration



# Value Iteration

## Value Iteration:

Find the optimal policy  $\pi_*$ :

- Given  $V_0(s)$ :
- Repeat:
  - For each  $s \in \mathcal{S}$ :
  - For each  $a \in \mathcal{A}(s)$ :
  - $Q(s, a) \leftarrow \sum_{(s', r) \in \mathcal{S} \times \mathcal{R}} p(s', r | s, a) [r + \gamma V_k(s')]$
  - $V_{k+1}(s) \leftarrow \max_{a \in \mathcal{A}(s)} Q(s, a)$

# Tutorial: Tic-Tac-Toe by Value Iteration

## Notes:

- 'x' goes first w.l.o.g.
- For 3x3 game, neither player can lose if they play optimally:
  - Do not train the AI, play dumb and see that it takes dumb move.
  - Do train the AI, play dumb, and lose to it.
  - Do Train the AI, play smart, and never win over it.

## Notes:

- $\mathcal{S} = \{1, -1, 0\}^9$
- $R_t = \begin{cases} 1, & \text{if } s_t \text{ win} \\ -1, & \text{if } s_t \text{ loses} \\ 0, & \text{otherwise} \end{cases}$
- $p(s', r | s, a) = \begin{cases} \frac{1}{|\text{legal}(s')|}, & \text{if } (s', r) \text{ is possible} \\ 0, & \text{otherwise} \end{cases}$
- Transition: ...

O	O	
	X	X
	O	

$s$

O	O	X
	X	X
	O	

$s, a$

O	O	X
X	X	X
	O	

O	O	X
	X	X
X	O	

O	O	X
	X	X
	O	X

$\{s'\}$

In the context of RL...

- Agent, environment, observations, state, reward, action, value, return, discount ...
- Evaluation, Iteration, Improvement, Value Iteration ...
- Monte Carlo, Off-policy
- Temporal Difference, Q-learning, Sarsa
- Function Approximation
- Policy Gradient Methods

# Monte Carlo Methods

- In real world, most of the time we have imperfect knowledge → estimate.
- Monte Carlo methods are *model-free*

# Monte Carlo Evaluation

- Goal: Given the *data acquired under  $\pi$* , estimate  $q_\pi$ .
- Approach: Express  $q_\pi$ -estimation problem as  $v_\pi$ -estimation problem,
  - Define a new problem where:

$$\bar{S}_t = (S_t, A_t)$$

→ Estimating  $v(\bar{s})$  is equivalent to estimating  $q_\pi(s, a)$ .

- Data =  $\{H_m = (\bar{s}_0, \bar{s}_1, \dots, \bar{s}_{T_m}), m = 1 \dots M\}$ .  
→ *Markov Reward Process*.
-

# Monte Carlo Evaluation

- Goal: Given the *data acquired under  $\pi$* , estimate  $q_\pi$ .
- Approach: Express  $q_\pi$ -estimation problem as  $v_\pi$ -estimation problem,
  - Define a new problem where:

$$\bar{S}_t = (S_t, A_t)$$

→ Estimating  $v(\bar{s})$  is equivalent to estimating  $q_\pi(s, a)$ .

- Data =  $\{H_m = (\bar{s}_0, \bar{s}_1, \dots, \bar{s}_{T_m}), m = 1 \dots M\}$ .  
→ *Markov Reward Process*.

- 
- Idea: Use averages to approximate  $v_\pi(s) \approx V(s)$ :
    - Batch update:

$$v_\pi(s) = E_\pi(G_t | S_t = s) \approx \frac{1}{C(s)} \sum_{m=1}^M \sum_{\tau=0}^{T_m-1} \mathbb{I}[s_\tau^m = s] g_\tau^m \triangleq V(s)$$



# Monte Carlo Evaluation

- Goal: Given the *data acquired under  $\pi$* , estimate  $q_\pi$ .
- Approach: Express  $q_\pi$ -estimation problem as  $v_\pi$ -estimation problem,
  - Define a new problem where:

$$\bar{S}_t = (S_t, A_t)$$

→ Estimating  $v(\bar{s})$  is equivalent to estimating  $q_\pi(s, a)$ .

- Data =  $\{H_m = (\bar{s}_0, \bar{s}_1, \dots, \bar{s}_{T_m}), m = 1 \dots M\}$ .  
→ *Markov Reward Process*.

- 
- Idea: Use averages to approximate  $v_\pi(s) \approx V(s)$ :

- Batch update:

$$v_\pi(s) = E_\pi(G_t | S_t = s) \approx \frac{1}{C(s)} \sum_{m=1}^M \sum_{\tau=0}^{T_m-1} \mathbb{I}[s_\tau^m = s] g_\tau^m \triangleq V(s)$$

- Iterative update after the  $m$ -th sample:

$$V(s_t^m) \leftarrow V(s_t^m) + \frac{1}{C(s_t^m)} (g_t^m - V(s_t^m))$$

- Or simply use a constant step size:

$$V(s_t^m) \leftarrow V(s_t^m) + \alpha (g_t^m - V(s_t^m))$$

$$\begin{aligned} \mu_k &= \frac{1}{k} \sum_{j=1}^k x_j \\ &= \frac{1}{k} \left( x_k + \sum_{j=1}^{k-1} x_j \right) \\ &= \frac{1}{k} (x_k + (k-1)\mu_{k-1}) \\ &= \mu_{k-1} + \frac{1}{k} (x_k - \mu_{k-1}) \end{aligned}$$

# MC Control

## Constant- $\alpha$ MC for estimating $\pi \approx \pi^*$

Algorithm inputs:

$\epsilon$

$\alpha$

$M$

Initialize arbitrarily:

$\pi \leftarrow$  some  $\epsilon$ -soft policy

$Q(s, a) \leftarrow$  some value for  $s \in \mathcal{S}, a \in \mathcal{A}(s)$

For  $m = 1, \dots, M$ :

Under  $\pi$  sample:  $s_0^m, a_0^m, r_1^m \dots a_{T_m-1}^m, r_{T_m}^m$

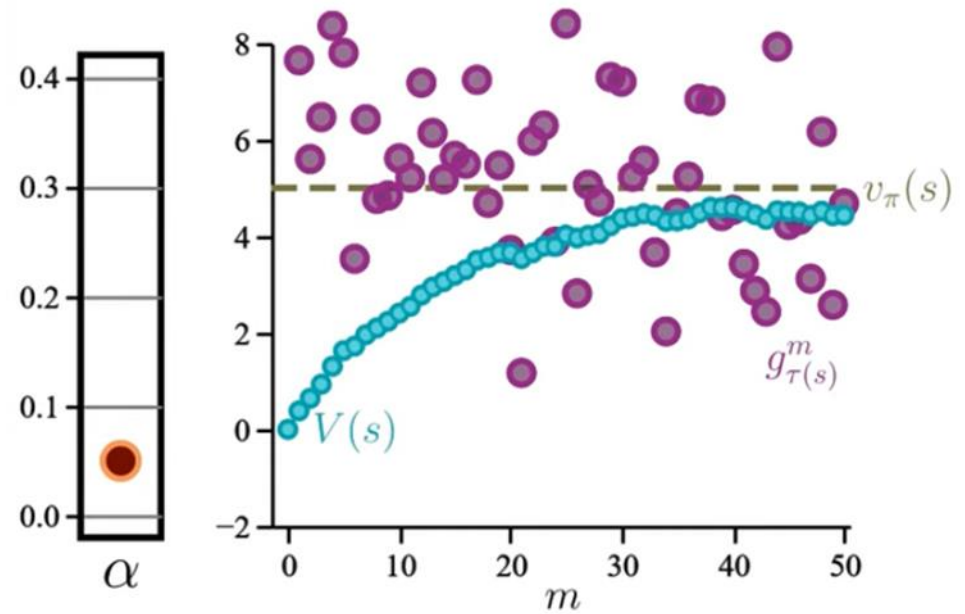
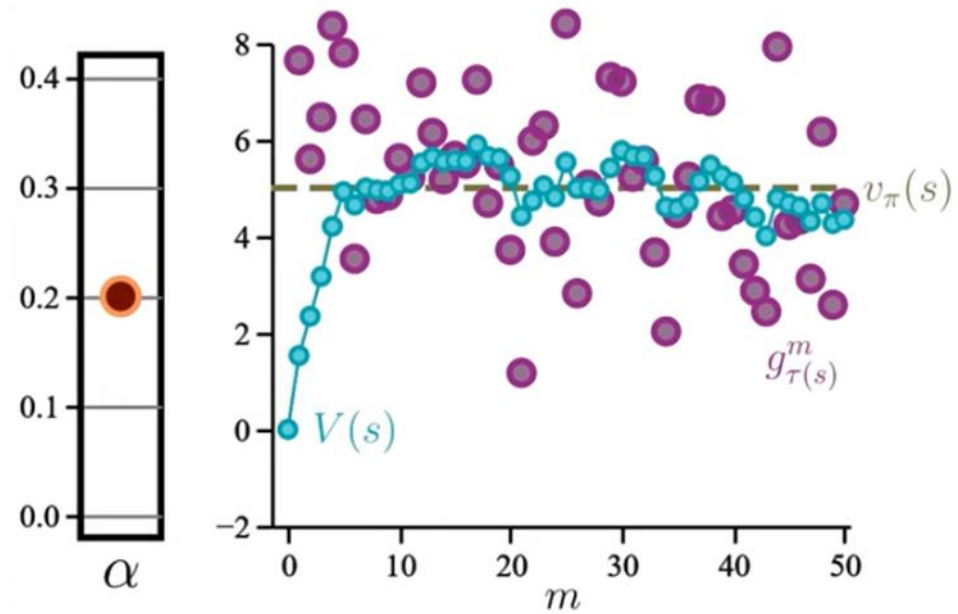
For  $t = 0, \dots, T_m - 1$ :

$g_t^m \leftarrow r_{t+1}^m + \gamma r_{t+2}^m + \dots$

$Q(s_t^m, a_t^m) \leftarrow Q(s_t^m, a_t^m) + \alpha(g_t^m - Q(s_t^m, a_t^m))$

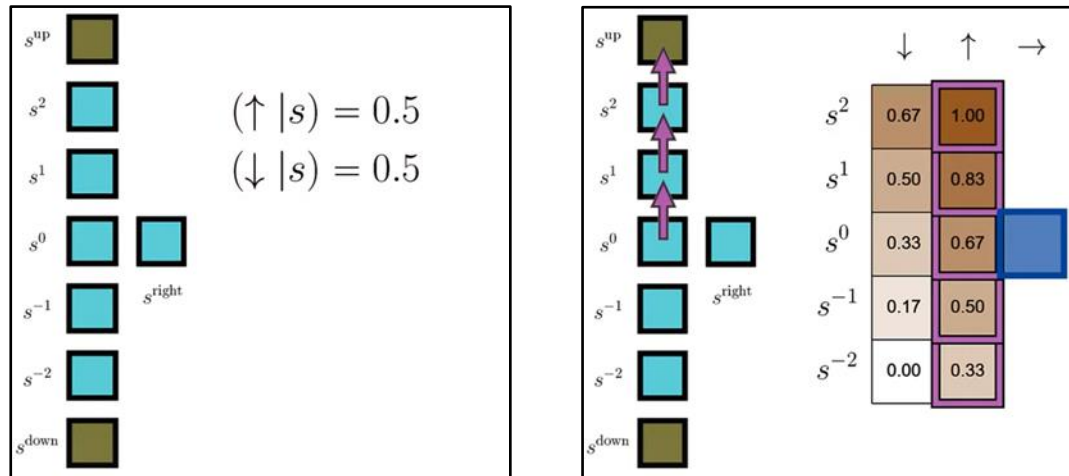
$\pi \leftarrow \epsilon$ -greedy( $Q$ )

# Monte Carlo Evaluation



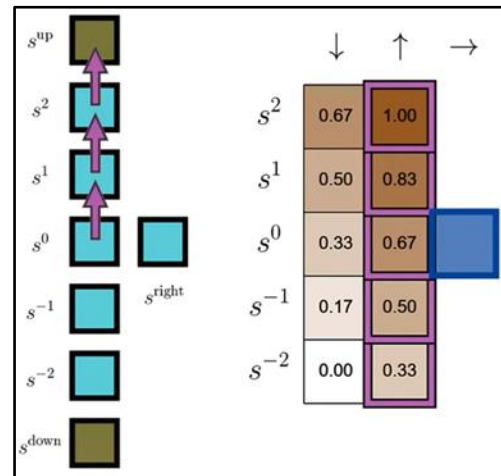
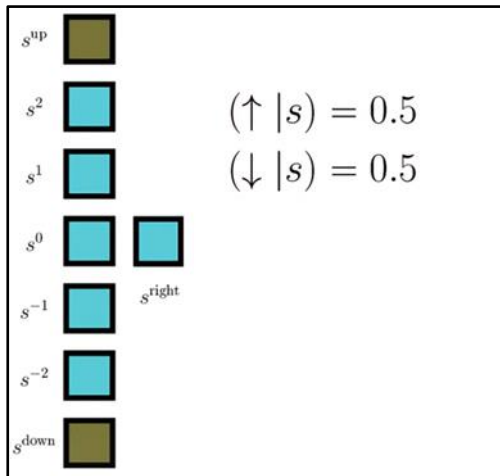
# Caveats of MC

- Trajectories have to terminate
- Exploration-Exploitation *dichotomy*:
  - To discover optimal policies, we must **explore** all state-action pairs.
  - To get high returns we must **exploit** known high state-action pairs.



# Caveats of MC

- Trajectories have to terminate
- Exploration-Exploitation dichotomy:
  - To discover optimal policies, we must **explore** all state-action pairs.
  - To get high returns we must **exploit** known high state-action pairs.



With infinite data,  $\pi_*$  is always discoverable if the policy is **SOFT**:

$$\pi(a|s) > 0 \quad \forall s \in \mathcal{S} \quad \forall a \in \mathcal{A}(s)$$

**$\epsilon$ -GREEDY POLICY OF  $Q$ :** With probability  $\epsilon$ , take an action selected uniformly from  $\mathcal{A}(s)$ , otherwise take  $\text{argmax}_a Q(s, a)$ .

# Off-policy method

- Goal:

Estimate  $q_{\pi}(s, a) = E_{\pi}[G_t | S_t = s, A_t = a]$

- Issue:

Data exists but collected under  $b$

- Remedy:

$$q_{\pi}(s, a) = E_b \left[ \frac{p_{\pi}(G_t)}{p_b(G_t)} G_t | S_t = s, A_t = a \right]$$

$$\rho = \prod_{\tau=t+1}^{T-1} \frac{\pi(A_{\tau} | S_{\tau})}{b(A_{\tau} | S_{\tau})}$$

$$\pi(a, s) > 0 \Rightarrow b(a, s) > 0$$

**BEHAVIOR POLICY:** Generates the data:

$$b(a|s)$$

**TARGET POLICY:** To be improved/evaluated:

$$\pi(a|s)$$

**ON-POLICY  
METHODS**

$$b = \pi$$

**OFF-POLICY  
METHODS**

$$b \neq \pi$$

# Off-policy MC

## Constant- $\alpha$ MC for estimating $\pi \approx \pi^*$

Algorithm inputs:

$\epsilon$                        $\alpha$                        $M$

Initialize arbitrarily:

$\pi \leftarrow$  some  $\epsilon$ -soft policy

$Q(s, a) \leftarrow$  some value for  $s \in \mathcal{S}$ ,  $a \in \mathcal{A}(s)$

For  $m = 1, \dots, M$ :

Under  $\pi$  sample:  $s_0^m, a_0^m, r_1^m \dots a_{T_m-1}^m, r_{T_m}^m$

For  $t = 0, \dots, T_m - 1$ :

$g_t^m \leftarrow r_{t+1}^m + \gamma r_{t+2}^m + \dots$

$Q(s_t^m, a_t^m) \leftarrow Q(s_t^m, a_t^m) + \alpha(g_t^m - Q(s_t^m, a_t^m))$

$\pi \leftarrow \epsilon$ -greedy( $Q$ )

## Off-Policy Constant- $\alpha$ MC for $\pi \approx \pi^*$

Algorithm inputs:

$b$                        $\alpha \in (0, 1]$                        $M \in \mathbb{N}$

Initialize arbitrarily:

$\pi \leftarrow$  some policy

$Q(s, a) \leftarrow$  some value for  $s \in \mathcal{S}$ ,  $a \in \mathcal{A}(s)$

For  $m = 1, \dots, M$ :

Under  $b$  sample:  $s_0^m, a_0^m, r_1^m \dots a_{T_m-1}^m, r_{T_m}^m$

For  $t = 0, \dots, T_m - 1$ :

$\rho_t^m \leftarrow \prod_{\tau=t+1}^{T_m-1} \frac{\pi(a_\tau^m | s_\tau^m)}{b(a_\tau^m | s_\tau^m)}$  (or 1 if  $t+1 > T_m - 1$ )

$g_t^m \leftarrow \rho_t^m (r_{t+1}^m + \gamma r_{t+2}^m + \dots)$

$Q(s_t^m, a_t^m) \leftarrow Q(s_t^m, a_t^m) + \alpha(g_t^m - Q(s_t^m, a_t^m))$

$\pi(s_t^m) \leftarrow \operatorname{argmax}_a Q(s_t^m, a)$  (ties broken arbitrarily)

In the context of RL...

- Agent, environment, observations, state, reward, action, value, return, discount ...
- Evaluation, Iteration, Improvement, Value Iteration ...
- Monte Carlo, Off-policy
- Temporal Difference, Q-learning, Sarsa
- Function Approximation
- Policy Gradient Methods



# Temporal Difference Learning

- *a priori:*


$$q_{\pi}(s, a) = E_{\pi}[\textcolor{red}{G}_t | (S_t, A_t) = (s, a)] = E_{\pi}[\textcolor{red}{R}_{t+1} + \gamma q_{\pi}(\textcolor{red}{S}_{t+1}, \textcolor{red}{A}_{t+1}) | (S_t, A_t) = (s, a)]$$

- *Just read through the maths:*  $Q(s, a) \approx g_t \approx r_t + \gamma Q(s_{t+1}, a_{t+1})$

- MC approach:

$$g_t^m = r_{t+1}^m + \gamma r_{t+2}^m \dots \gamma^{T_m-1} r_{T_m}^m$$

target


$$Q(s_t^m, a_t^m) \leftarrow Q(s_t^m, a_t^m) + \alpha(g_t^m - Q(s_t^m, a_t^m))$$

# Temporal Difference Learning

- *a priori*:

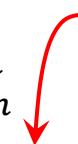
$$q_{\pi}(s, a) = E_{\pi}[\textcolor{red}{G}_t | (S_t, A_t) = (s, a)] = E_{\pi}[\textcolor{red}{R}_{t+1} + \gamma q_{\pi}(\textcolor{red}{S}_{t+1}, \textcolor{red}{A}_{t+1}) | (S_t, A_t) = (s, a)]$$

- *Just read through the maths:*  $Q(s, a) \approx g_t \approx r_t + \gamma Q(s_{t+1}, a_{t+1})$

- MC approach:

$$g_t^m = r_{t+1}^m + \gamma r_{t+2}^m \dots \gamma^{T_m-1} r_{T_m}^m$$

target

$$Q(s_t^m, a_t^m) \leftarrow Q(s_t^m, a_t^m) + \alpha(g_t^m - Q(s_t^m, a_t^m))$$




- 1-step TD approach:

$$\hat{g}_t^m = r_{t+1}^m + \gamma \boxed{Q(s_{t+1}, a_{t+1})}$$

bootstrap

$$Q(s_t^m, a_t^m) \leftarrow Q(s_t^m, a_t^m) + \alpha(\hat{g}_t^m - Q(s_t^m, a_t^m))$$

target



# SARSA

## Constant- $\alpha$ MC for estimating $\pi \approx \pi^*$

Algorithm inputs:

$\epsilon$                        $\alpha$                        $M$

Initialize arbitrarily:

$\pi \leftarrow$  some  $\epsilon$ -soft policy

$Q(s, a) \leftarrow$  some value for  $s \in \mathcal{S}, a \in \mathcal{A}(s)$

For  $m = 1, \dots, M$ :

Under  $\pi$  sample:  $s_0^m, a_0^m, r_1^m \dots a_{T_m-1}^m, r_{T_m}^m$

For  $t = 0, \dots, T_m - 1$ :

$g_t^m \leftarrow r_{t+1}^m + \gamma r_{t+2}^m + \dots$

$Q(s_t^m, a_t^m) \leftarrow Q(s_t^m, a_t^m) + \alpha(g_t^m - Q(s_t^m, a_t^m))$

$\pi \leftarrow \epsilon$ -greedy( $Q$ )

## On Policy TD Control: n-step SARSA

Changes:

- Approximating the rewards beyond the  $n$ -th step with the current value of  $Q(s, a)$  (bootstrapping):

$$g_{t:t+n}^m = r_{t+1}^m + \dots + \gamma^{n-1} r_{t+n}^m + \gamma^n Q(s_{t+n}^m, a_{t+n}^m)$$

$$Q(s_t^m, a_t^m) \leftarrow Q(s_t^m, a_t^m) + \alpha(g_{t:t+n}^m - Q(s_t^m, a_t^m))$$

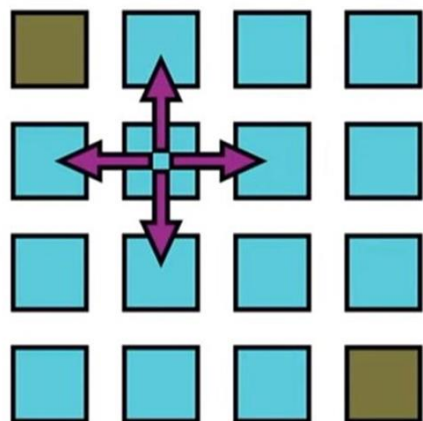
- Updates happen *during* the episode, Interweaving between  $(S, A, R)$  tuples, with an  $n$  step delay.
- The policy is updated in similar manner with MC

# TD $\ni$ MC

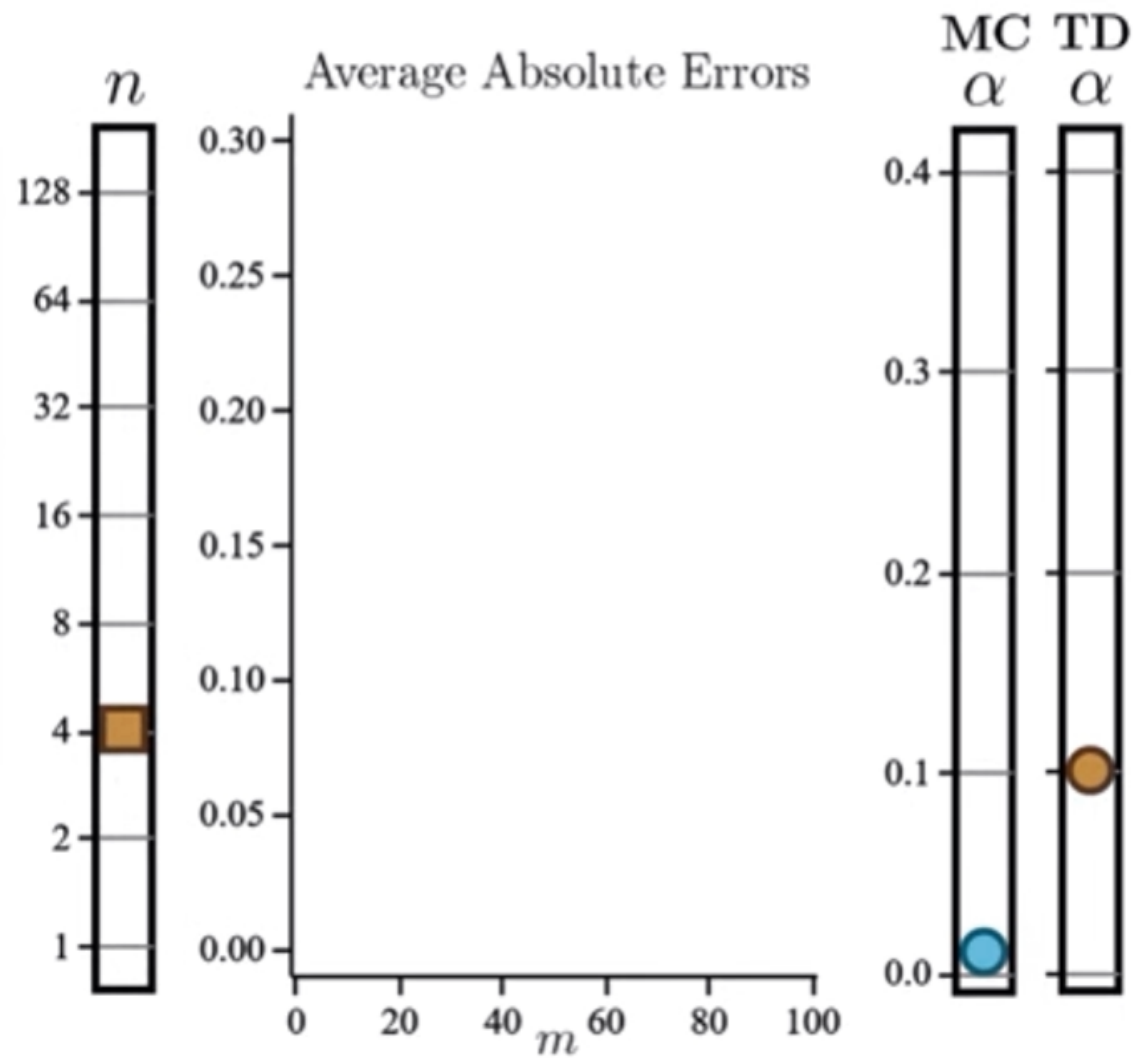
## Evaluation Example: MC vs TD

# States = 11

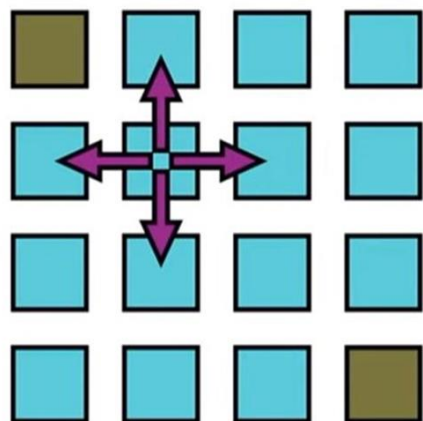
# Algo Runs = 200



$R_t = -1$   
 $\pi(a|s) = 0.25$



# TD $\ni$ MC



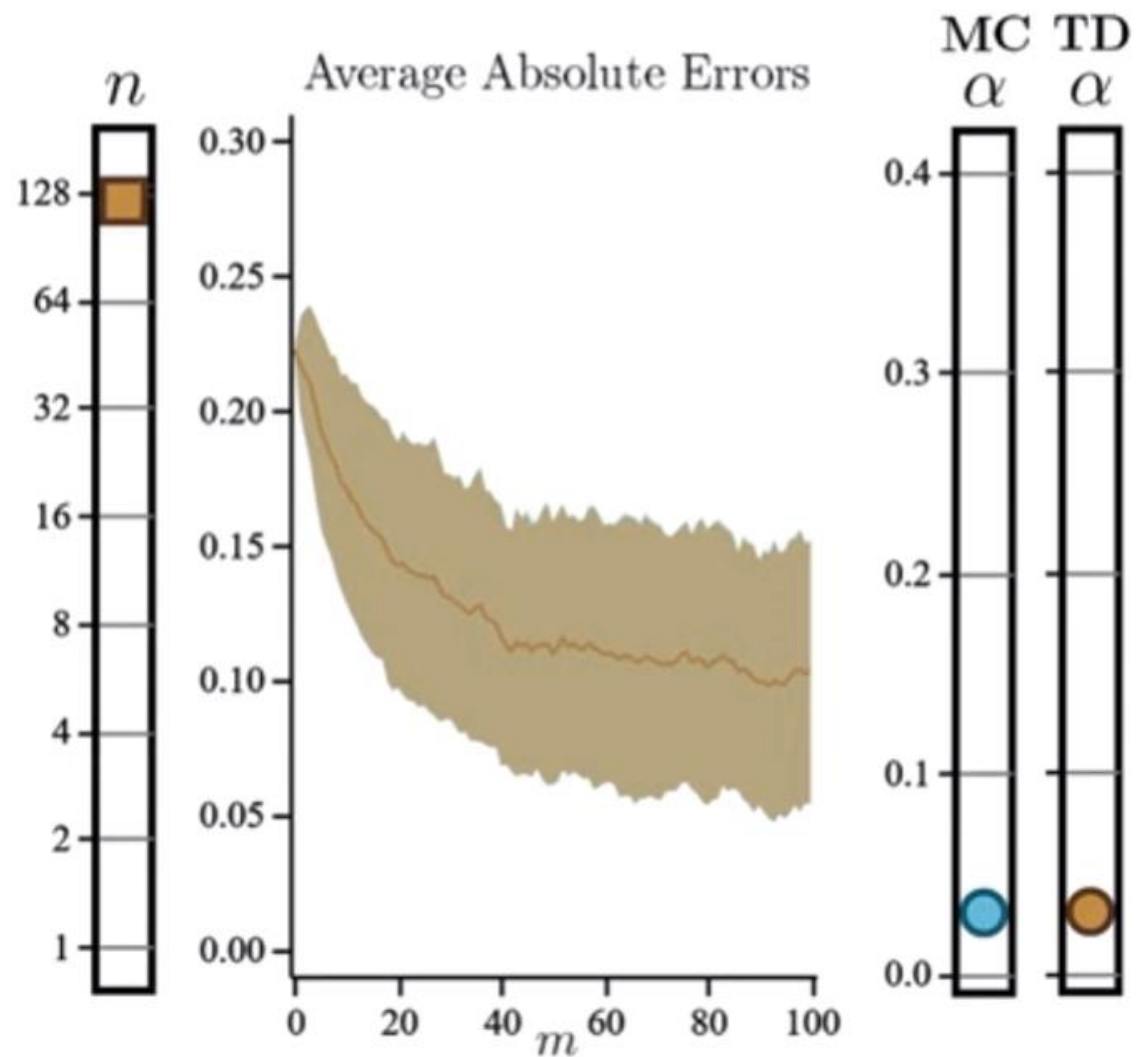
$$R_t = -1$$

$$\pi(a|s) = 0.25$$

## Evaluation Example: MC vs TD

# States = 11

# Algo Runs = 200



# Q-learning

### Constant- $\alpha$ MC for estimating $\pi \approx \pi^*$

Algorithm inputs:

M

Initialize arbitrarily:

 $\pi \leftarrow \text{some } \epsilon\text{-soft policy}$ 
$$Q(s, a) \leftarrow \text{some value for } s \in \mathcal{S}, a \in \mathcal{A}(s)$$

For  $m = 1, \dots, M$ :

Under  $\pi$  sample:  $s_0^m, a_0^m, r_1^m \cdots a_{T_m-1}^m, r_{T_m}^m$

For  $t = 0, \dots, T_m - 1$ :

$$g_t^m \leftarrow r_{t+1}^m + \gamma r_{t+2}^m + \dots$$
$$Q(s_t^m, a_t^m) \leftarrow Q(s_t^m, a_t^m) + \alpha(g_t^m - Q(s_t^m, a_t^m))$$
$$\pi \leftarrow \epsilon\text{-greedy}(Q)$$

## Q-Learning

From 1-step **TD** Control, the primary adjustment is to the target:

$$\begin{array}{c} r_{t+1}^m + \gamma Q(s_{t+1}^m, a_{t+1}^m) \\ \downarrow \\ r_{t+1}^m + \gamma \max_a Q(s_{t+1}^m, a) \end{array}$$

The max operator means this is **off-policy**.

Under the behavior policy, we are targeting  $q_*$ .

There's also a change to the update's timing:

1-step **TD**:      update  $Q$       update  $Q$       update  $Q$   
 $s_0^m, a_0^m, r_1^m, s_1^m, a_1^m, r_2^m, s_2^m, a_2^m, r_3^m, s_3^m, a_3^m, r_4^m \dots$   
1-step **Q**:      update  $Q$       update  $Q$       update  $Q$

In the context of RL...

- Agent, environment, observations, state, reward, action, value, return, discount ...
- Evaluation, Iteration, Improvement, Value Iteration ...
- Monte Carlo, Off-policy
- Temporal Difference, Q-learning, Sarsa
- **Function Approximation**
- Policy Gradient Methods

# Function Approximation

- When  $\mathcal{S}$  is continuous  $\rightarrow$  never enough data.

- Example:

- Assume  $v_\pi(s) = \hat{v}(s, w)$ ,  $\hat{v}(s + \delta, w) = \hat{v}(s, w) + \left(\frac{\partial \hat{v}}{\partial s}\right)^\top \delta$ .

- Goal:

$$\min_w \sum_{s \in \{s_i\}} \|v_\pi(s_i) - \hat{v}(s_i, w)\|^2$$

- Update rule:

$$w \leftarrow w + \alpha [G_i - \hat{v}(s_i, w)] \nabla_w \hat{v}(s_i, w)$$

$$\nabla_w \hat{v}(s_i, w) = \frac{\partial \hat{v}}{\partial w}$$

- DRL  $\rightarrow w$  is param of DNN.



# Function Approximation

- Example:
  - Assume  $q_\pi(s) = \hat{q}(s, a, w)$
  - Goal:

$$\min_w \sum_{s \in \{s_i\}} \|q_\pi(s_i, a_i) - \hat{q}(s_i, a_i, w)\|^2$$

- Update rule:

$$w \leftarrow w + \alpha [G_i - \hat{q}(s_i, a_i, w)] \nabla_w \hat{q}(s_i, a_i, w)$$
$$\nabla_w \hat{q}(s_i, a_i, w) = \frac{\partial \hat{q}}{\partial w}$$

In the context of RL...

- Agent, environment, observations, state, reward, action, value, return, discount ...
- Evaluation, Iteration, Improvement, Value Iteration ...
- Monte Carlo, Off-policy
- Temporal Difference, Q-learning, Sarsa
- Function Approximation
- Policy Gradient Methods

# Policy Gradient Methods

## REINFORCE

To specify upfront:

- Functional form:  $\pi(a|s, \boldsymbol{\theta})$
- Initial  $\boldsymbol{\theta}$
- Step size  $\alpha$

For  $m = 1, \dots, M$ :

Sample:  $s_0^m, a_0^m, r_1^m \dots a_{T_m-1}^m, r_{T_m}^m$

For  $t = 0, \dots, T_m - 1$ :

$$g_t^m \leftarrow r_{t+1}^m + \gamma r_{t+2}^m + \dots$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t g_t^m \nabla \ln \pi(a_t^m | s_t^m, \boldsymbol{\theta})$$

- $\nabla_{\boldsymbol{\theta}} \ln \pi(a_t | s_t, \boldsymbol{\theta})$  gives the “direction” that increasing  $\boldsymbol{\theta}$  will increase  $\pi(a_t | s_t, \boldsymbol{\theta})$ .

$$\nabla \ln \pi(a_t^m | s_t^m, \boldsymbol{\theta}) = \frac{\nabla \pi(a_t^m | s_t^m, \boldsymbol{\theta})}{\pi(a_t^m | s_t^m, \boldsymbol{\theta})}$$

# Policy Gradient Methods

## REINFORCE

To specify upfront:

- Functional form:  $\pi(a|s, \theta)$
- Initial  $\theta$
- Step size  $\alpha$

For  $m = 1, \dots, M$ :

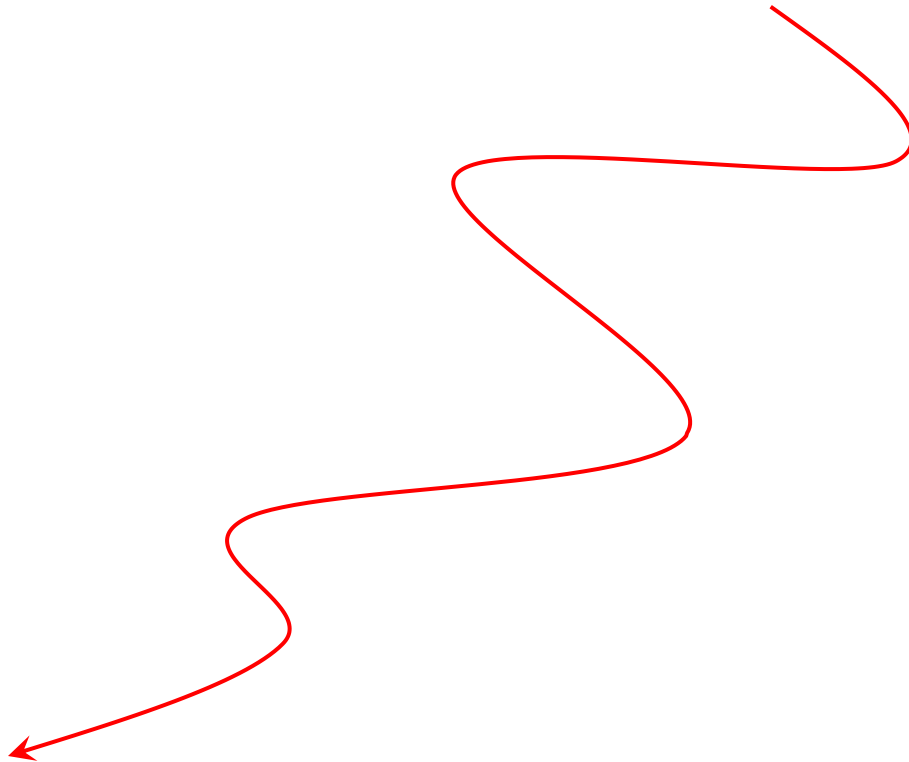
Sample:  $s_0^m, a_0^m, r_1^m \dots a_{T_m-1}^m, r_{T_m}^m$

For  $t = 0, \dots, T_m - 1$ :

$$g_t^m \leftarrow r_{t+1}^m + \gamma r_{t+2}^m + \dots$$

$$\theta \leftarrow \theta + \alpha \gamma^t g_t^m \nabla \ln \pi(a_t^m | s_t^m, \theta)$$

- $\nabla_{\theta} \ln \pi(a_t | s_t, \theta)$  gives the “direction” that increasing  $\theta$  will increase  $\pi(a_t | s_t, \theta)$ .
- The increase of  $\theta$  is  $\sim g_t \nabla_{\theta} \ln \pi(a_t | s_t, \theta)$



# Policy Gradient Methods

## REINFORCE

To specify upfront:

- Functional form:  $\pi(a|s, \theta)$
- Initial  $\theta$
- Step size  $\alpha$

For  $m = 1, \dots, M$ :

Sample:  $s_0^m, a_0^m, r_1^m \dots a_{T_m-1}^m, r_{T_m}^m$

For  $t = 0, \dots, T_m - 1$ :

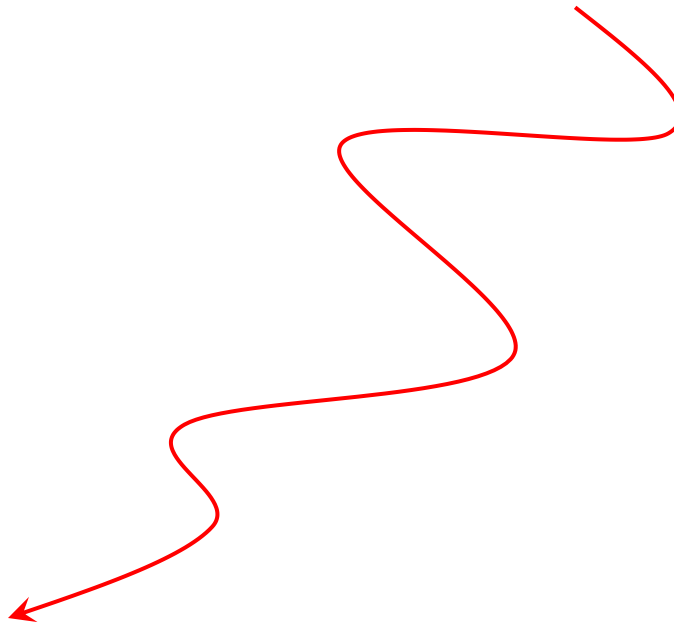
$$g_t^m \leftarrow r_{t+1}^m + \gamma r_{t+2}^m + \dots$$

$$\theta \leftarrow \theta + \alpha \gamma^t g_t^m \nabla \ln \pi(a_t^m | s_t^m, \theta)$$

- $\nabla_{\theta} \ln \pi(a_t | s_t, \theta)$  gives the “direction” that increasing  $\theta$  will increase  $\pi(a_t | s_t, \theta)$ .

- The increase of  $\theta$  is  $\sim g_t \nabla_{\theta} \ln \pi(a_t | s_t, \theta)$

→ the higher the return  $g_t$  an action  $a_t$  yields, the higher the probability of an action is *increased*.



# Policy Gradient Methods

- **Actor-Critic Methods** combine elements of policy-based methods and value-based methods.

- It introduces an advantage function

$$A(s_i, a_i) = Q(s, a) - V(s)$$

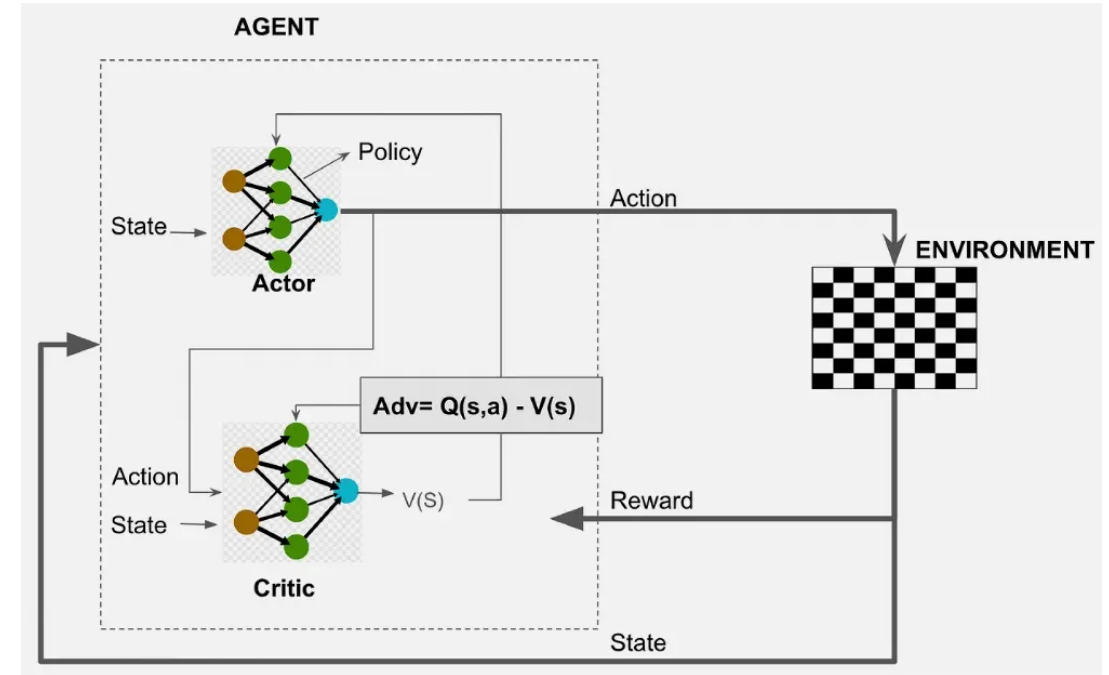
→ provides a measure of how “good” and action is compared with the average action.

- “Actor” gradient:

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_i \nabla_{\theta} \ln(\pi(a_i | s_i, \theta)) A(s_i, a_i)$$

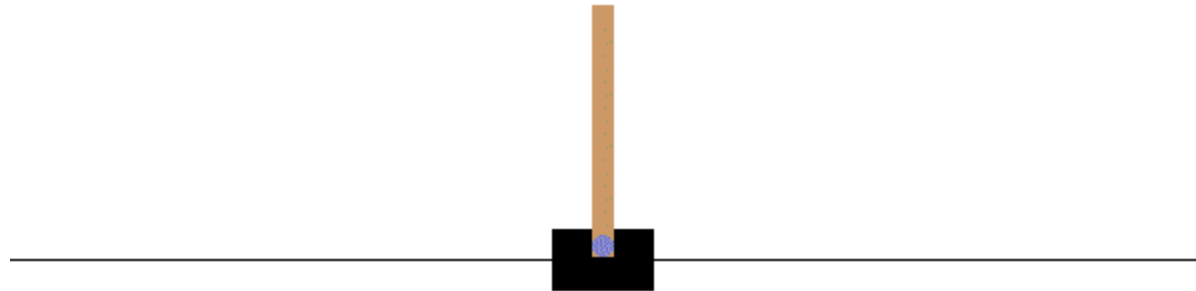
- “Critic” gradient:

$$\nabla_w J(w) \approx \frac{1}{N} \sum_i \nabla_w A(s_i, a_i)^2$$



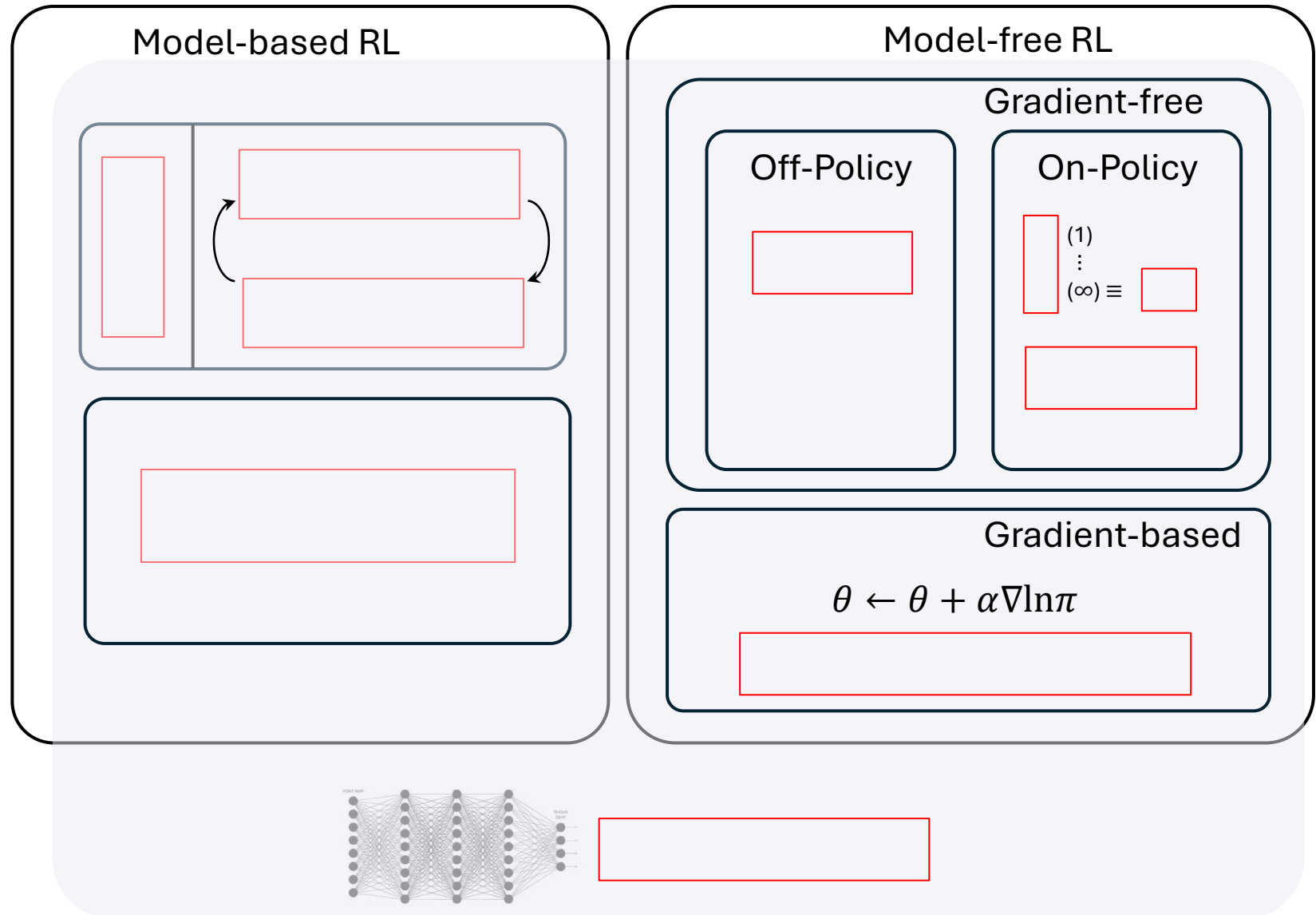
# Tutorial: CartPole

[https://www.gymnasium.dev/environments/classic\\_control/cart\\_pole/](https://www.gymnasium.dev/environments/classic_control/cart_pole/)



# Summary

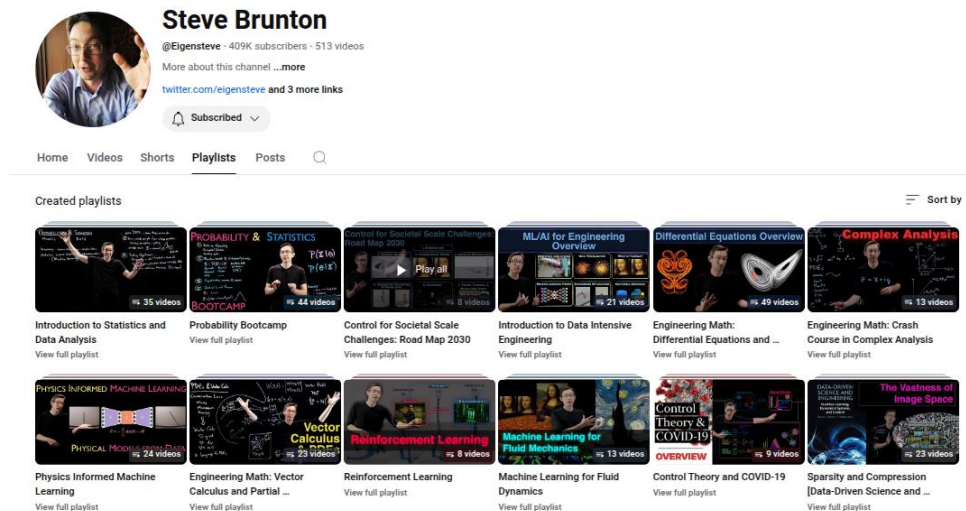
- Value Evaluation, Policy Iteration, Policy Improvement, Value Iteration...
- MC
- TD, Q-learning, Sarsa
- Function Approximation (Deep RL)
- Policy Gradient Methods





# References

- [Reinforcement Learning: An Introduction, Sutton](#)
- [David Silver RL Lectures](#)
- [Zhao Shiyu RL Lectures](#)
- [OpenAI Introduction to RL](#)



**Steve Brunton**  
@Eigensteve · 409K subscribers · 513 videos  
More about this channel ...more  
[twitter.com/eigensteve](#) and 3 more links  
Subscribed

Home Videos Shorts **Playlists** Posts

Created playlists Sort by

Introduction to Statistics and Data Analysis View full playlist

Probability Bootcamp View full playlist

Control for Societal Scale Challenges: Road Map 2030 View full playlist

ML/AI for Engineering Overview View full playlist

Differential Equations Overview View full playlist

Complex Analysis View full playlist

Physics Informed Machine Learning View full playlist

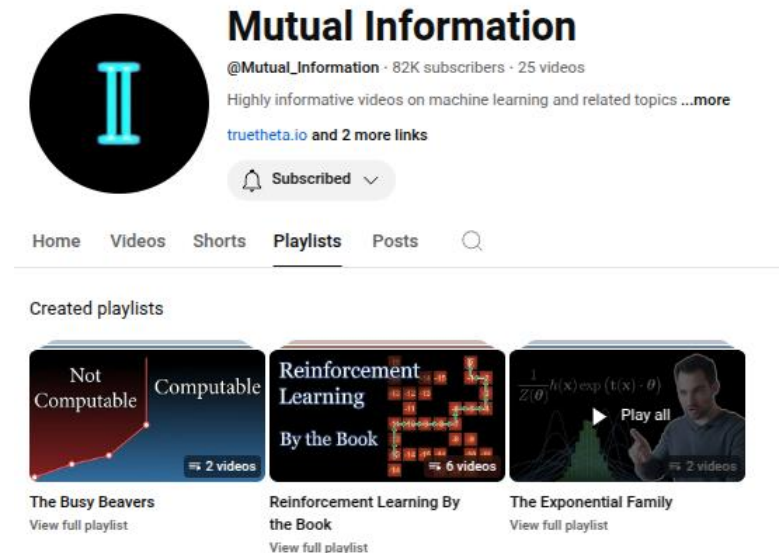
Engineering Math: Vector Calculus and Partial ... View full playlist

Reinforcement Learning View full playlist

Machine Learning for Fluid Dynamics View full playlist

Control Theory and COVID-19 View full playlist

Sparsity and Compression [Data-Driven Science and ...] View full playlist



**Mutual Information**  
@Mutual\_Information · 82K subscribers · 25 videos  
Highly informative videos on machine learning and related topics ...more  
[truetheta.io](#) and 2 more links  
Subscribed

Home Videos Shorts **Playlists** Posts

Created playlists

The Busy Beavers View full playlist

Reinforcement Learning By the Book View full playlist

The Exponential Family View full playlist

