Introduction To Reinforcement Learning

Thien-Minh Nguyen, PhD
Centre for Advance Robotics Technology Innovation

April 2025

Outline

- Recent progresses in RL.
- Overview of RL
- Basic methods to solve the RL problems
- Tutorials
    - Tic-tac-toe. Complete MDP with *Value Iteration* method.
    - Cartpole. Small scaled DRL 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:

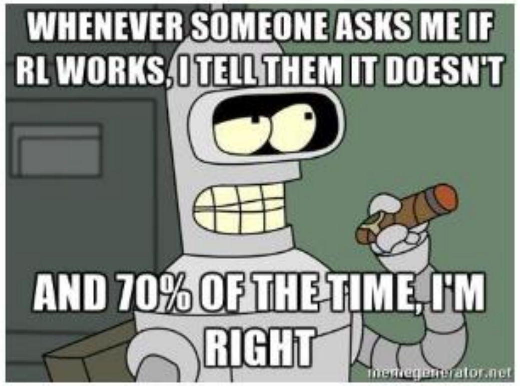
```
@misc{rlblogpost,
    title={Deep Reinforcement Learning Doesn't Work Yet},
    author={Irpan, Alex},
    howpublished={\url{https://www.alexirpan.com/2018/02/14/rl-hard.html}},
    year={2018}
}
```

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.



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

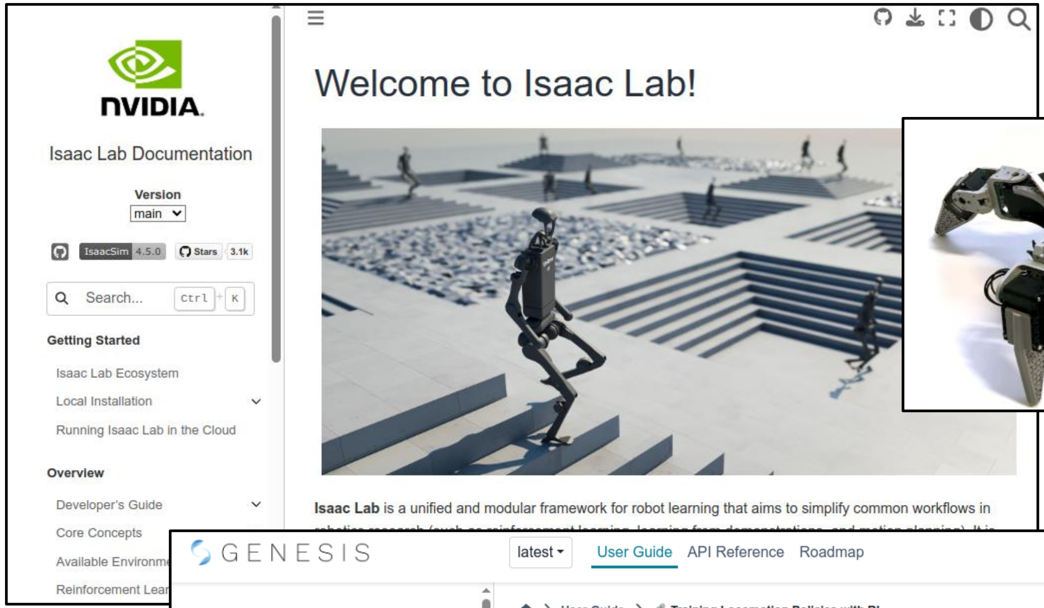
https://www.alexirpan.com/2018/02/14/rl-hard.html

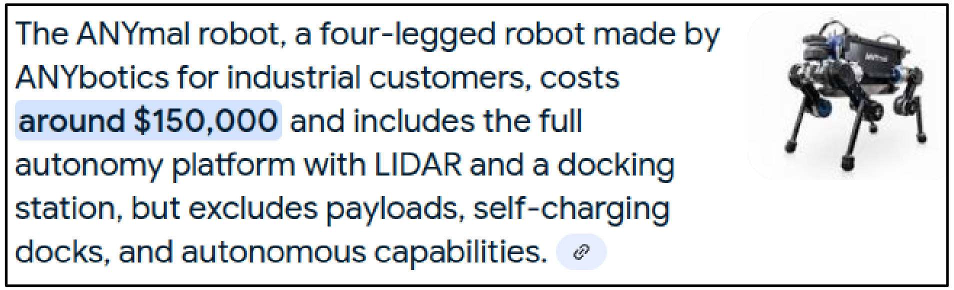
Why RL ain't work?

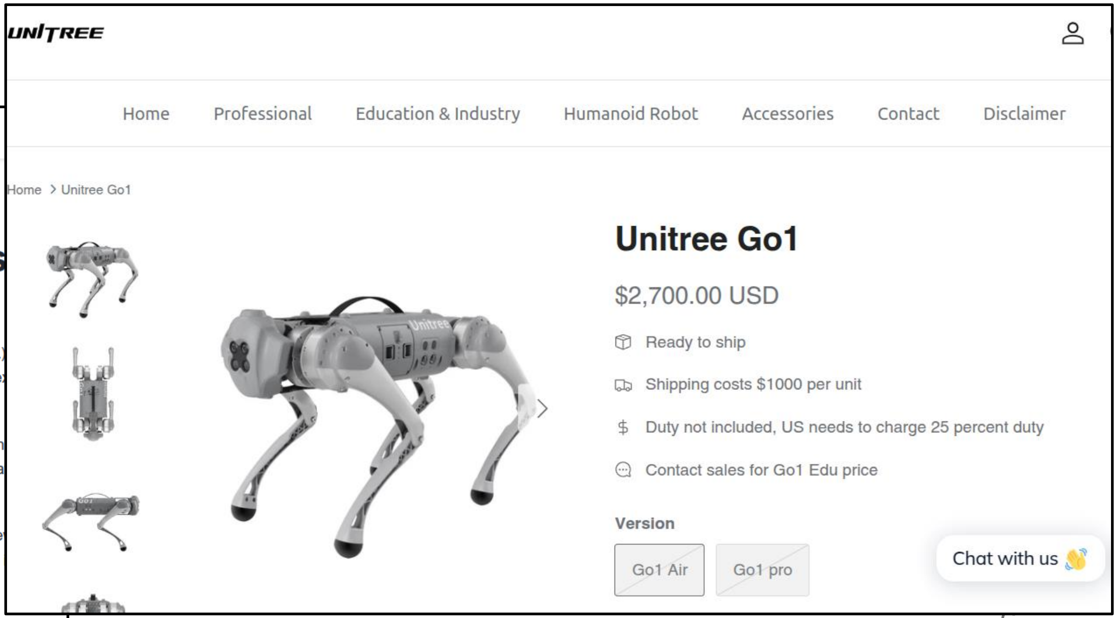
- Sample Inefficient
- Can be solved by other methods
- Always requires a reward function
- Reward function design is difficult
- Local optima hard to escape
- Overfitting
- Unstable and hard to reproduce

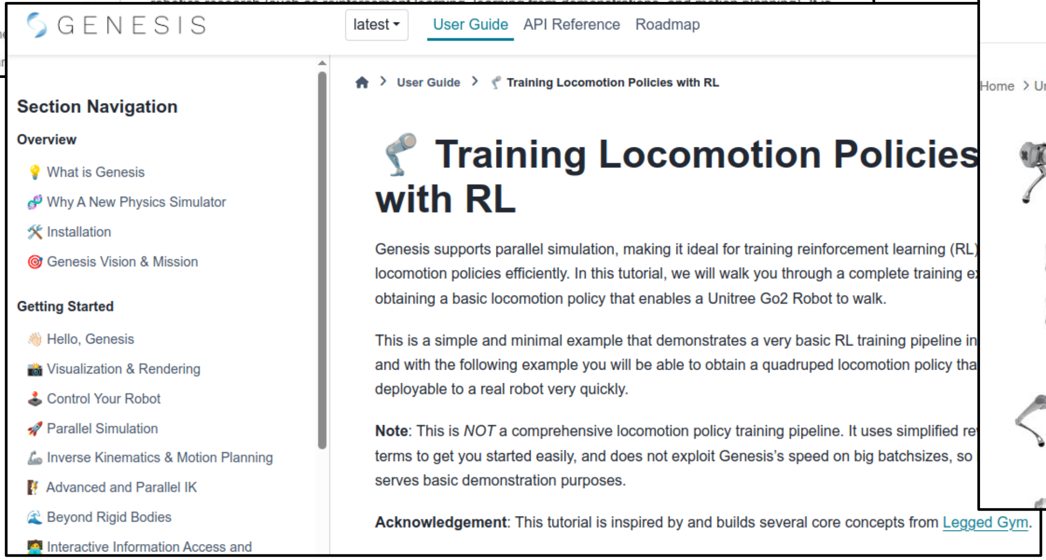
Why RL *works* now?

- Sample Inefficient → Cost of experiment ↓



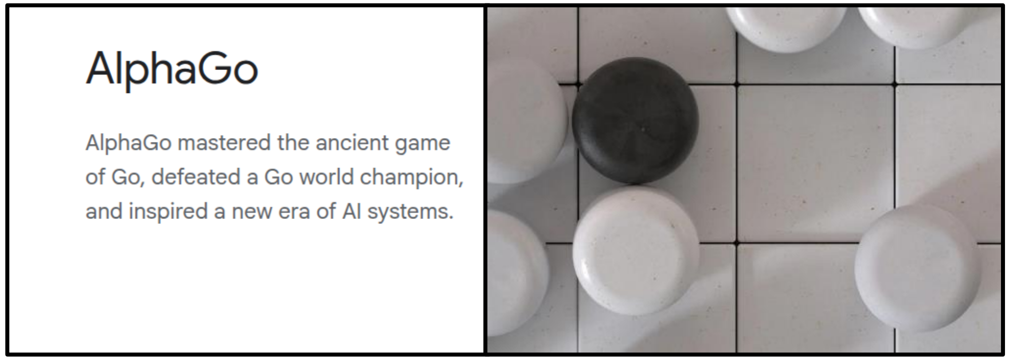






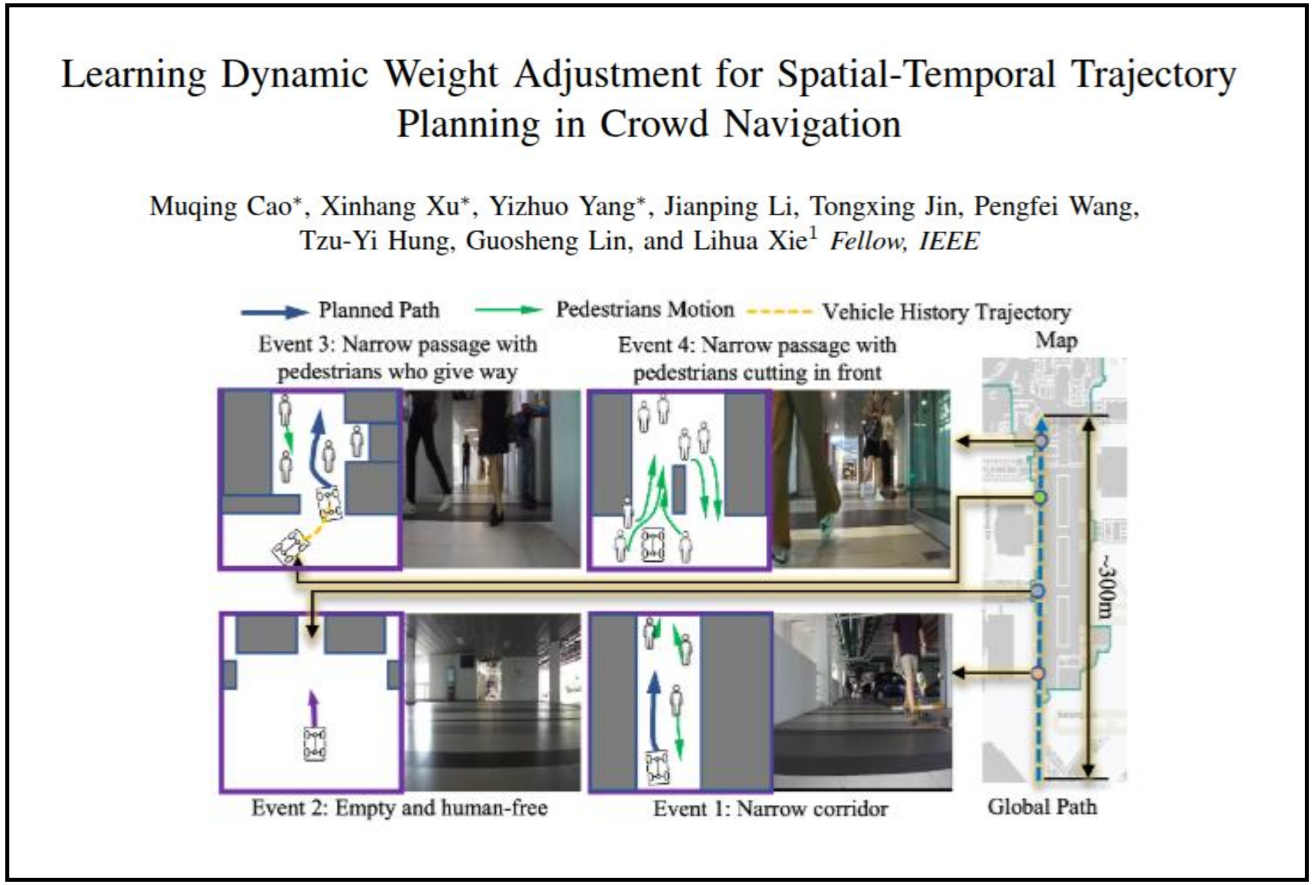
Why RL *works* now?

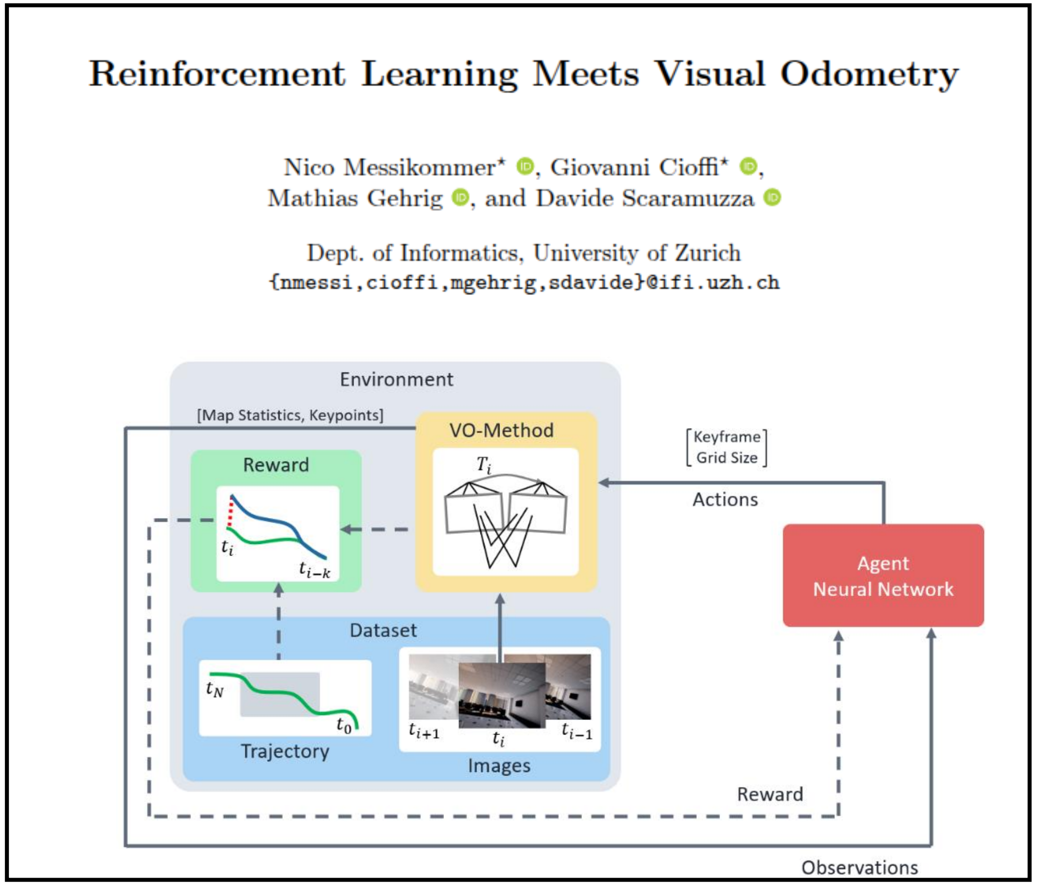
- Sample Inefficient → **Cost of experiment** ↓
- Some problems can be solved by other methods → and many others can be solved by RL



Why RL *works* now?

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



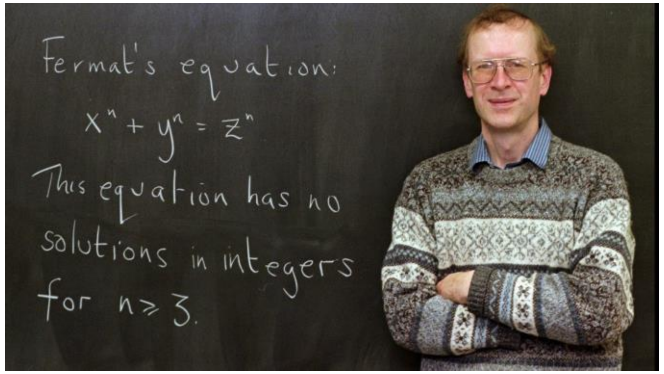


Why RL *works* now?

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

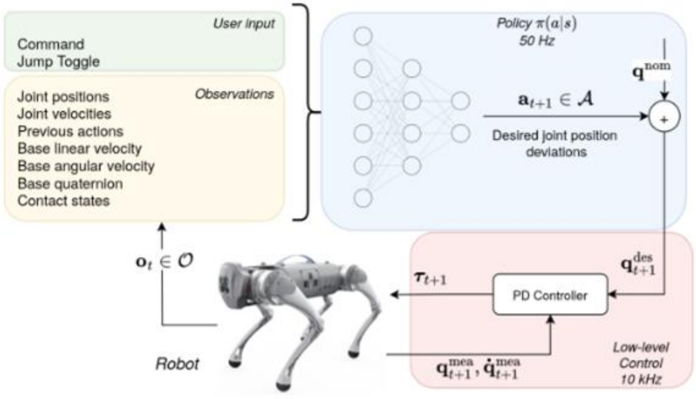


Curriculum-Based Reinforcement Learning for Quadrupedal Jumping: A Reference-free Design

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. $w_\times$ IS THE WEIGHT, $\sigma_\times$ IS A SCALING FACTOR FOR THE EXPONENTIAL KERNEL, $e(\cdot)$ AND $\log(\cdot)$ SEPARATELY DENOTE THE EXPONENT AND LOGARITHM OPERATION.

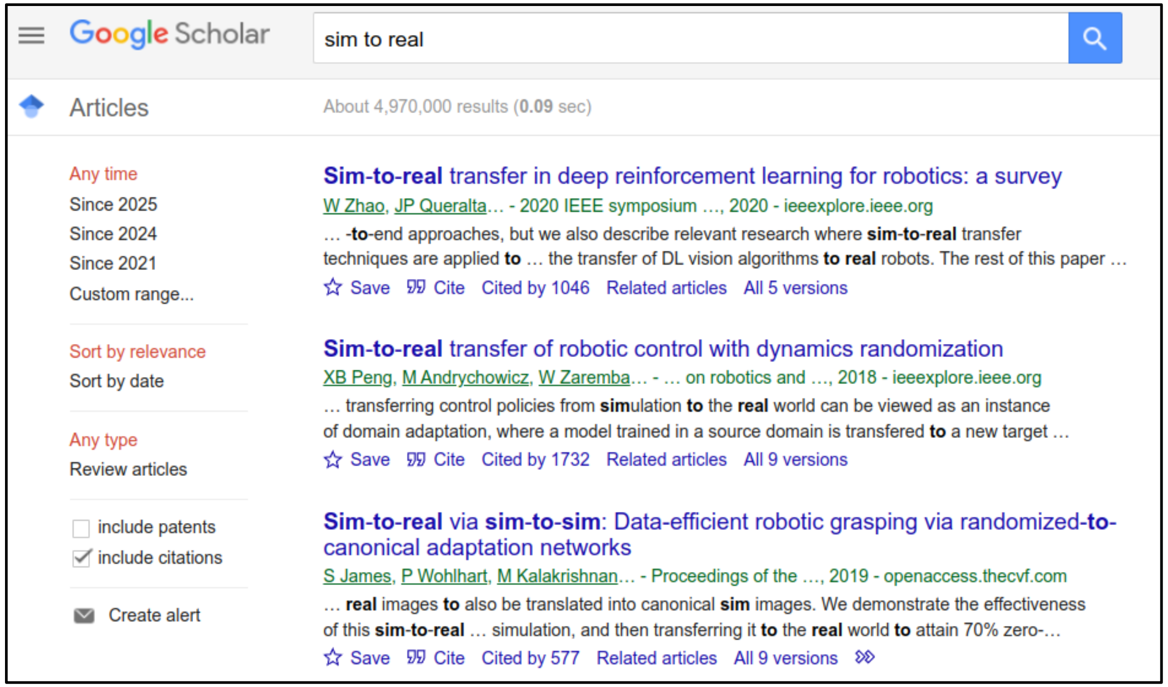
Name	Type	Stance	Flight	Landing
Landing position	Single	0	0	$w_\mathbf{p}(e(-\sum \|\|\mathbf{p}_{\text{land}} - \mathbf{p}_{\text{des}}\|\|^2)/\sigma_{p,\text{land}})$
Landing orientation	Single	0	0	$w_{\text{ori}}(e(-\|\|\log(\bar{\mathbf{q}}_{\text{land}}^{-1} * \bar{\mathbf{q}}_{\text{des}}\|\|^2)/\sigma_{\text{ori,land}})$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

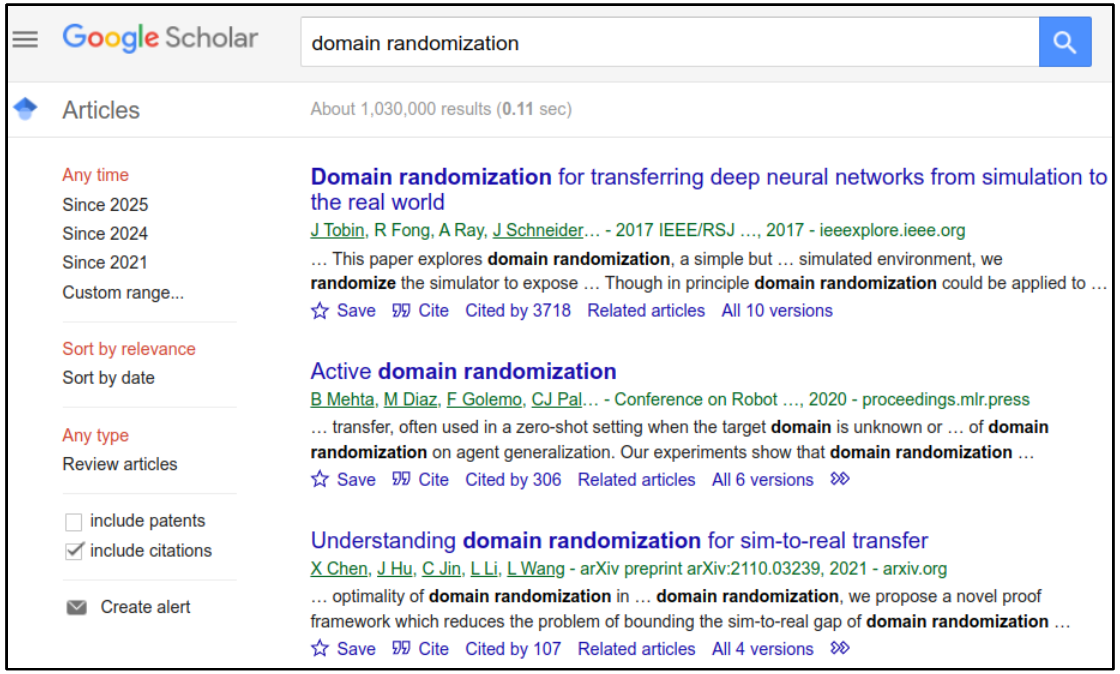


Why RL *works* now?

- 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**





Why RL *works* now?

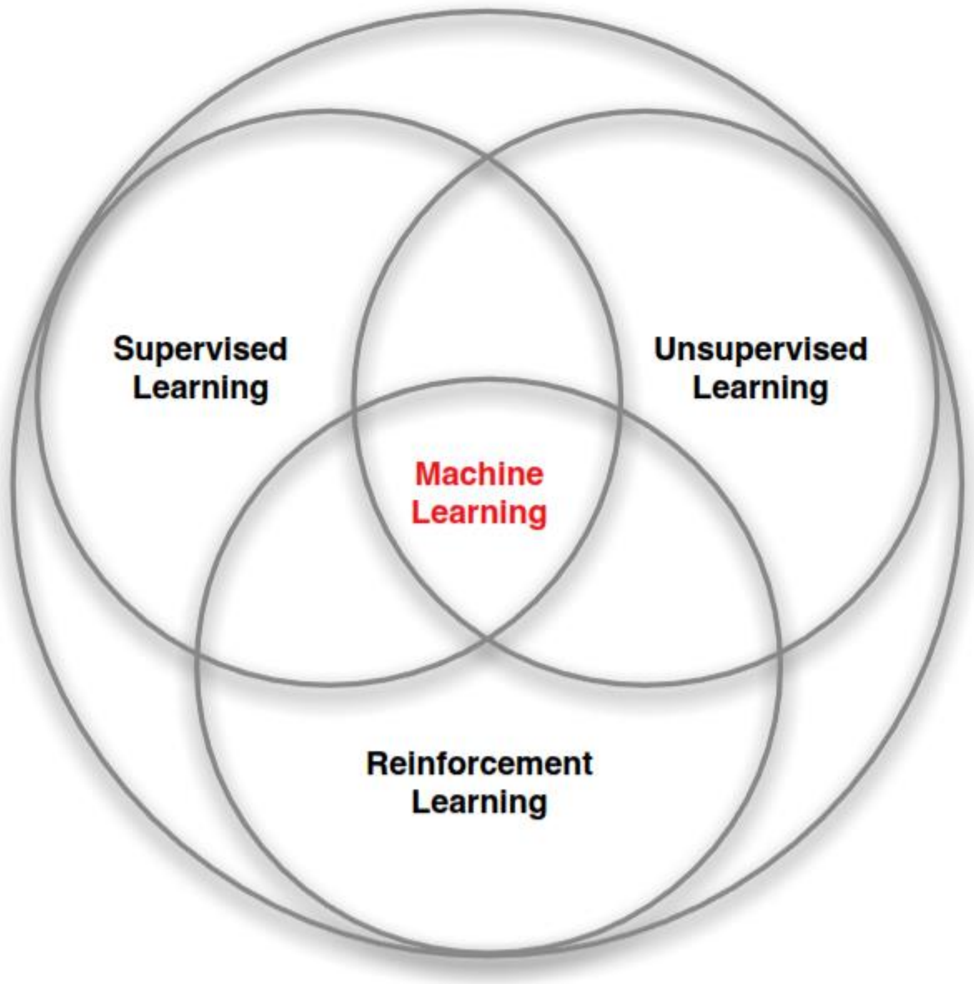
- Sample Inefficient → **Cost of experiment ↓**
- Some problems can be solved by other methods → **Some can be solved by RL**
- Always requires a reward function
- Reward function design is difficult
- Local optima hard to escape
- Overfitting
- Unstable and hard to reproduce

→ **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

Have labels of objects → make learning model that predicts the label of new objects

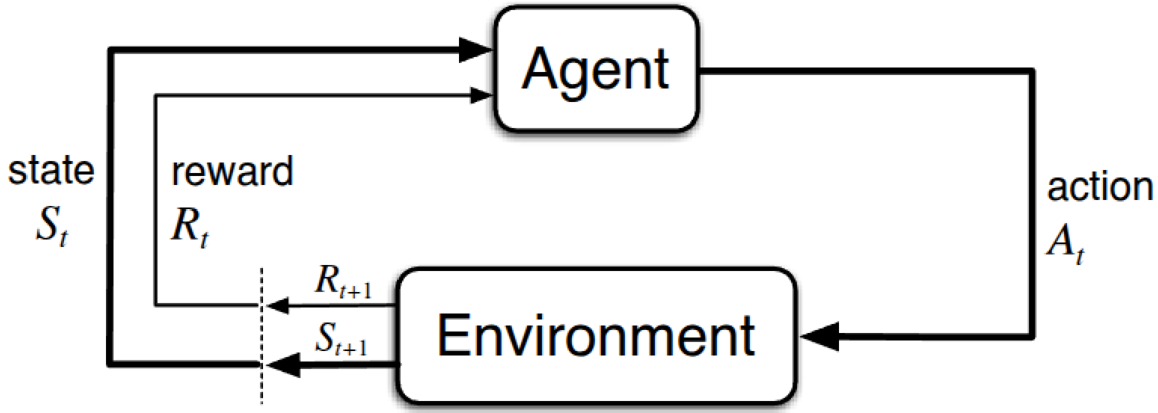


Have data with assumed correlation model → discovering the model

Have agent-environment dynamics → find the policy that yields optimal *return*.

Agent and Environment

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



The robot belongs to which category?

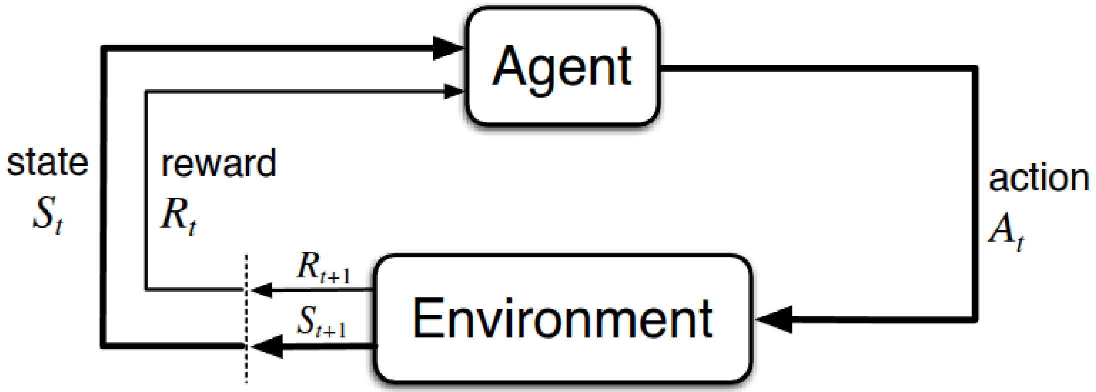
"All goals can be described by the maximization of expected cumulative reward"

main IsaacLab / source / isaaclab_tasks / isaaclab_tasks / direct / anymal_c / **anymal_c_env_cfg.py**

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

```python
class AnymalCFlatEnvCfg(DirectRLEnvCfg):
    decimation = 4
    action_scale = 0.5
    action_space = 12
    observation_space = 48
    state_space = 0

    # simulation
    sim: SimulationCfg = SimulationCfg( ...
    )
    terrain = TerrainImporterCfg( ...
    )

    # scene
    scene: InteractiveSceneCfg = InteractiveSceneCfg(num_envs=4096, env_spacing=4.0, replicate_physics=True)

    # events
    events: EventCfg = EventCfg()

    # robot
    robot: ArticulationCfg = ANYMAL_C_CFG.replace(prim_path="/World/envs/env_.*/Robot")
    contact_sensor: ContactSensorCfg = ContactSensorCfg(
        prim_path="/World/envs/env_.*/Robot/.*", history_length=3, update_period=0.005, track_air_time=True
    )

    # reward scales
    lin_vel_reward_scale = 1.0
    yaw_rate_reward_scale = 0.5
    z_vel_reward_scale = -2.0
    ang_vel_reward_scale = -0.05
    joint_torque_reward_scale = -2.5e-5
    joint_accel_reward_scale = -2.5e-7
    action_rate_reward_scale = -0.01
    feet_air_time_reward_scale = 0.5
    undesired_contact_reward_scale = -1.0
    flat_orientation_reward_scale = -5.0
```

Concepts (1)



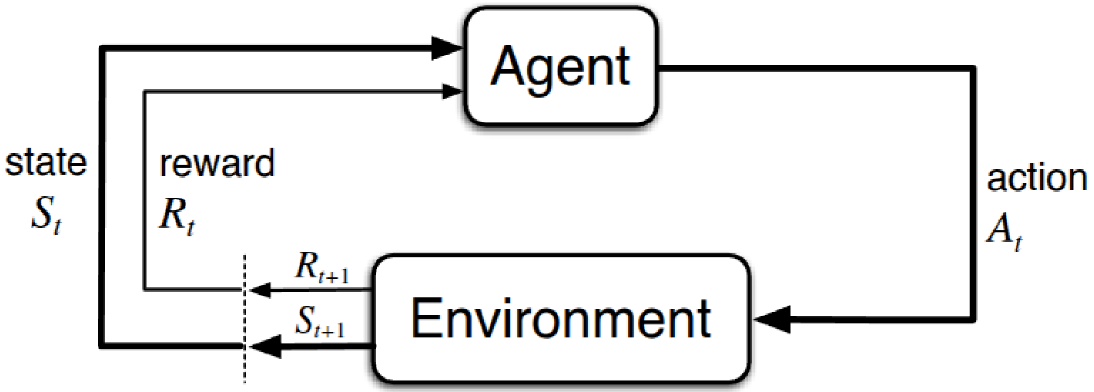
Definition

- 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.** Some states are called *terminal states*.
  - $t \in \{0, 1, 2 \ldots\}, s \in \mathcal{S}, a \in \mathcal{A}(s), r \in \mathcal{R} \subset \mathbb{R}$
  - The dynamics between agent and environment is summarized in:

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

- $H_t \triangleq (S_0, R_0, A_0 \ldots S_t, R_t, A_t)$, the **trajectory.**
- $O_t = h(H_t)$ is the **observation**, e.g., image, can be IMU reading, lidar scan ...
- Often, we need to estimate the state from the observation $S_t = f(O_t)$

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.** Some states are called *terminal states.*
  - $t \in \{0, 1, 2 \dots\}, s \in \mathcal{S}, a \in \mathcal{A}(s), r \in \mathcal{R} \subset \mathbb{R}$
  - The dynamics between agent and environment is summarized in:

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

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

Concepts (2) [1]

Definition

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

    $$a = \pi(s)$$

  - Stochastic policy:

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

- The return G_t:

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

- The discount factor:

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

Definition

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

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

- The action-value function:

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

Note

S_t, A_t, R_t, G_t are all random variables that can take value s, a, r, g in their respective domain

Somewhere in the multiverse…

- Jiedi Wan is an “RL researcher” at SpadeX, he runs an experiment and sends this data to his boss Yilong Ma:

$$o_0, a_0, r_0, o_1, a_1, r_1, o_2, a_2, r_2, o_3 \ldots, a_{t_{terminal}}$$

- Yilong fires Jiedi Wan…

https://tinyurl.com/tmnRL2025

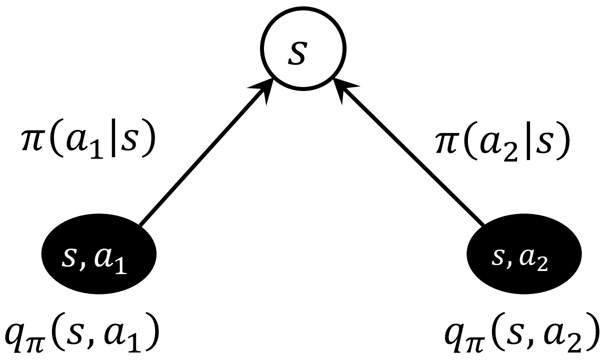
Bellman Equation

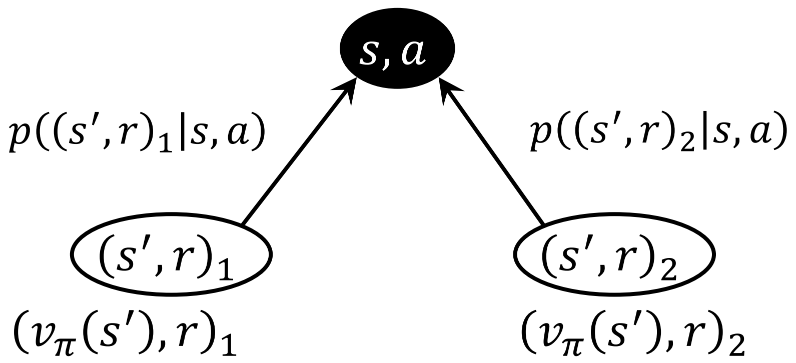
- For state-value function:

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

- For action-value function:

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





Bellman Equation

- For state-value function:

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

- For action-value function:

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

Optimal Value Functions & BOE

Definition

- $\pi > \pi' \Rightarrow v_\pi(s) > v_{\pi'}(s), \forall s$
- The optimal state-value function $v_*(s)$:

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

- The optimal action-value function $q_*(s,a)$:

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

- For any optimal π_*, all $s \in \mathcal{S}$, all $a \in \mathcal{A}(s)$:

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

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

Theorem

For any MDP:

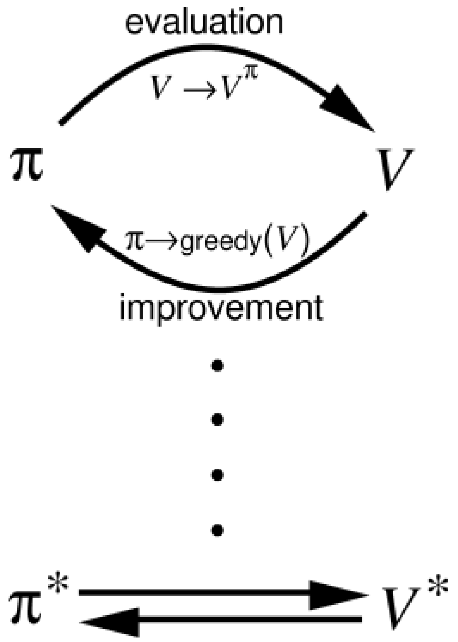
- $\exists \pi_*, \pi_* \geq \pi, \forall \pi$
- $v_{\pi_*}(s) = v_*(s), \forall \pi_*$
- $q_{\pi_*}(s,a) = q_\pi(s,a), \forall \pi_*$

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* $\pi \rightarrow \underbrace{\textit{evaluate } \pi}_{\substack{\textit{evaluation} \\ \text{(given } \pi \text{ find } v_\pi)}} \rightarrow \underbrace{\textit{improve } \pi}_{\substack{\textit{improvement} \\ \text{(given } v_\pi \text{find } \pi' \geq \pi)}}$*, 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 $k = 0 \dots K-1$:

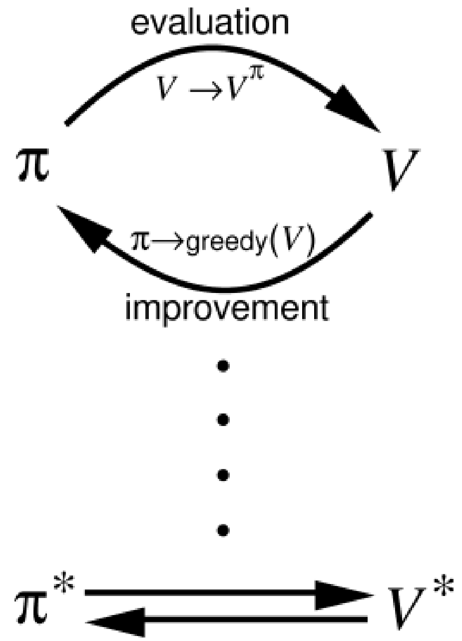
$$\forall s \in \mathcal{S}\colon\ V_{k+1}(s) \leftarrow \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{(s',r) \in \mathcal{S} \times \mathcal{R}} p(s',r|s,a)[r + \gamma V_k(s')]\,,$$

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

Policy *Improvement*:

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

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



Policy *Iteration*

Solving the MDP

Policy *Evaluation*:

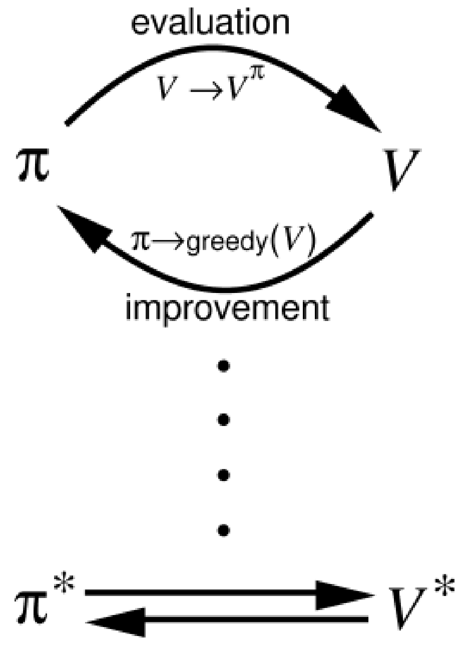
Given a policy $\pi(a|s)$
- For $k = 0 \ldots K-1$:

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

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

Policy *Improvement*:

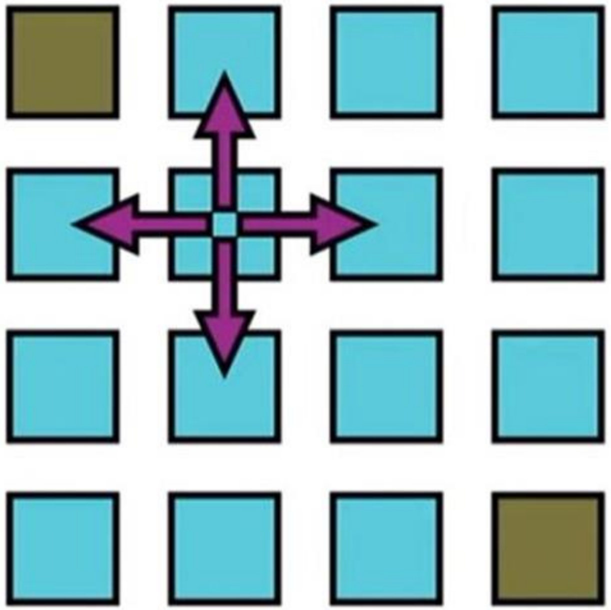
Given a value function $v_\pi(s)$:
- $\pi_* = \text{greedy}(v_\pi(s))$



Policy *Iteration*

Policy Evaluation – Grid World Example

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

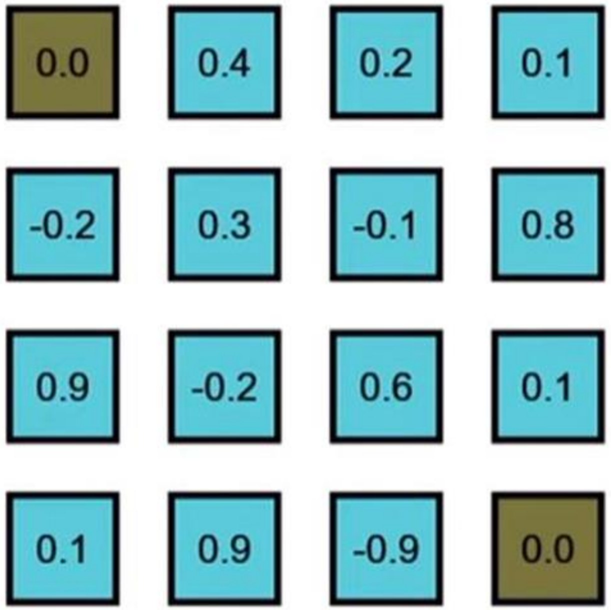


$R_t = -1$

$\pi(a|s) = 0.25$

Policy Evaluation – Grid World Example

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

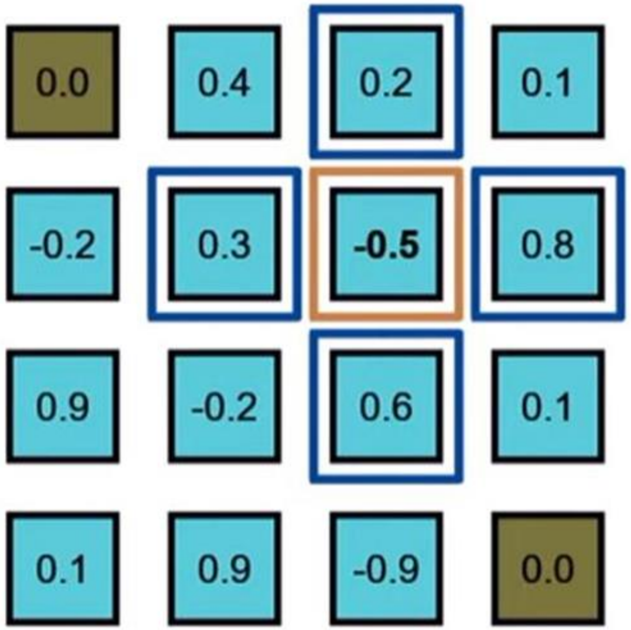


$$R_t = -1$$

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

Policy Evaluation – Grid World Example

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



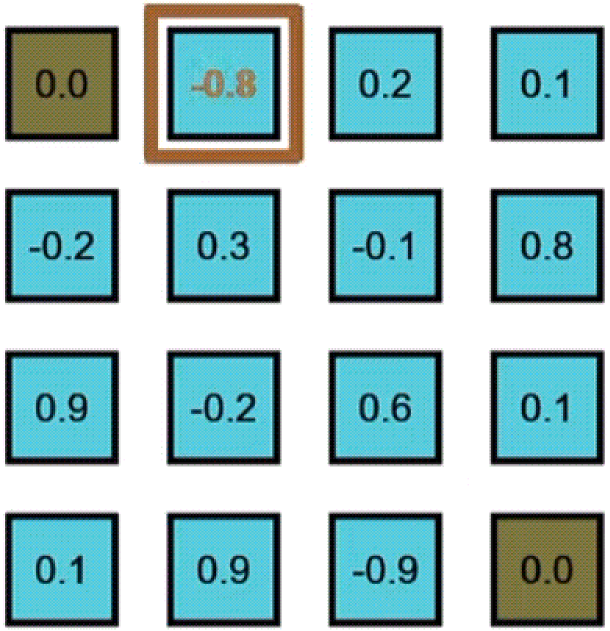
$R_t = -1$

$\pi(a|s) = 0.25$

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

Policy Evaluation – Grid World Example

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

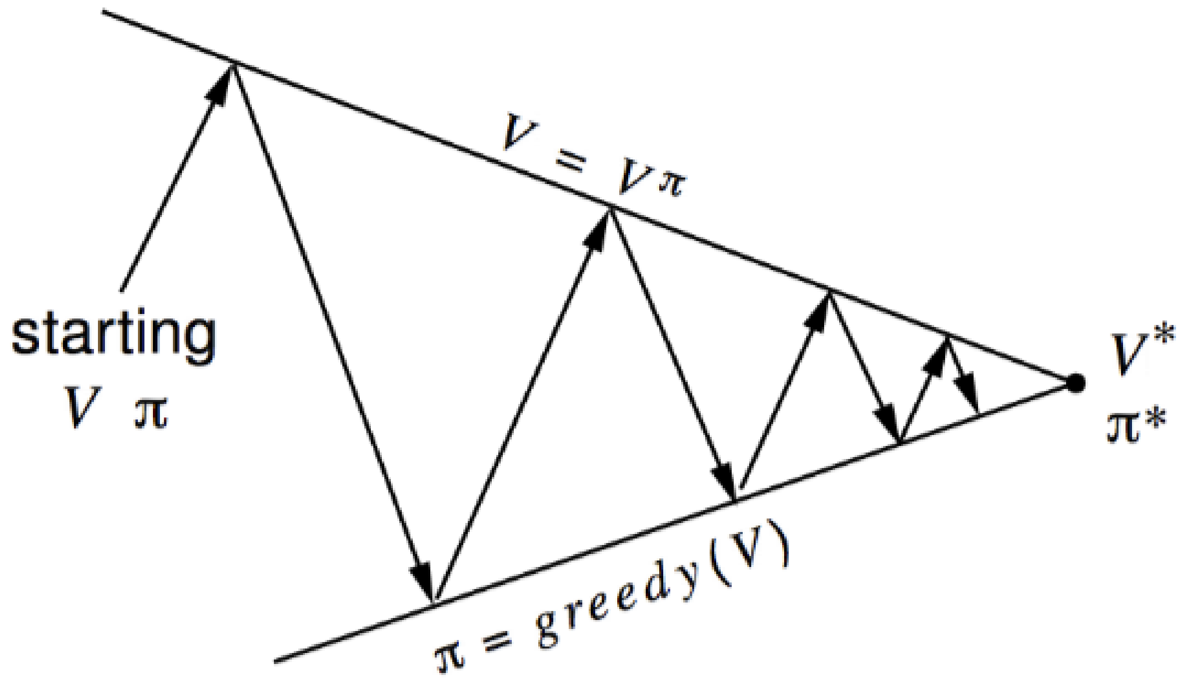


$$R_t = -1$$

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

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

Policy Iteration



Value *Iteration*

Value *Iteration*:

Find the optimal policy π_*:

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

Or by invoking the state-action value

Value *Iteration*:

Find the optimal policy π_*:

- 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:

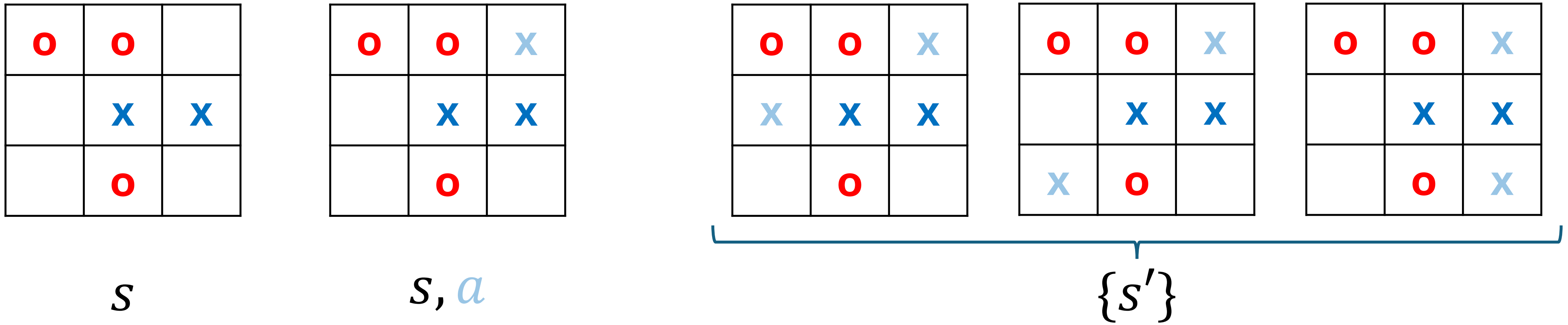
  → Do not train the AI, play dumb and see that it takes dumb move.

  → 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: ...



https://tinyurl.com/tmnDRL2025TTT

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* π, estimate q_π.
- Approach: Express q_π-estimation problem as v_π-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 = $\left\{H_m = \left(\bar{s}_0, \bar{s}_1, \dots \bar{s}_{T_m}\right), m = 1 \dots M\right\}$.

    → *Markov Reward Process*.

Monte Carlo Evaluation

- Goal: Given the *data acquired under* π, estimate q_π.
- Approach: Express q_π-estimation problem as v_π-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* π, estimate q_π.
- Approach: Express q_π-estimation problem as v_π-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:
  $$v_\pi(s) = E_\pi(G_t | S_t = s) \approx \frac{1}{C(s)} \sum_{m=1}^{M} \sum_{\tau=0}^{T_m - 1} \mathbb{I}[s_\tau^m = s]\, g_\tau^m \triangleq V($$
  - Iterative update after the m-th sample:
  $$V(s_t^m) \leftarrow V(s_t^m) + \frac{1}{C(s_t^m)}\left(g_t^m - V(s_t^m)\right)$$
  - Or simply use a constant step size:
  $$V(s_t^m) \leftarrow V(s_t^m) + \alpha\left(g_t^m - V(s_t^m)\right)$$

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

MC Control

Constant-α MC for estimating $\pi \approx \pi*$

Algorithm inputs:

ϵ α M

Initialize arbitrarily:

$\pi \leftarrow$ some ϵ-soft policy

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

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

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

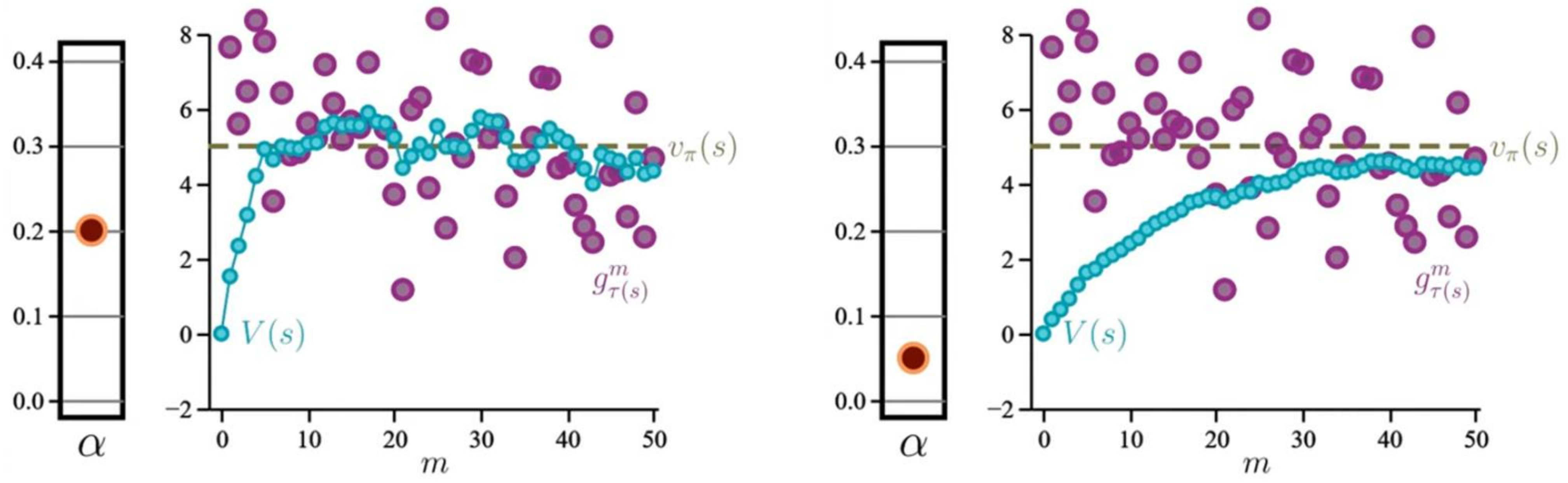
For $t = 0, \cdots, T_m - 1$:

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

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

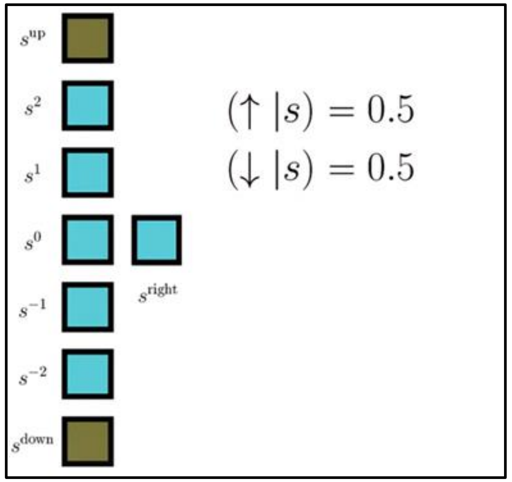
$\pi \leftarrow \epsilon\text{-greedy}(Q)$

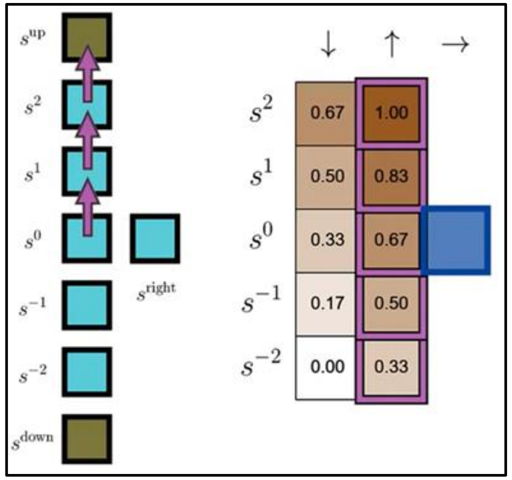
Monte Carlo Evaluation



Caveats of MC

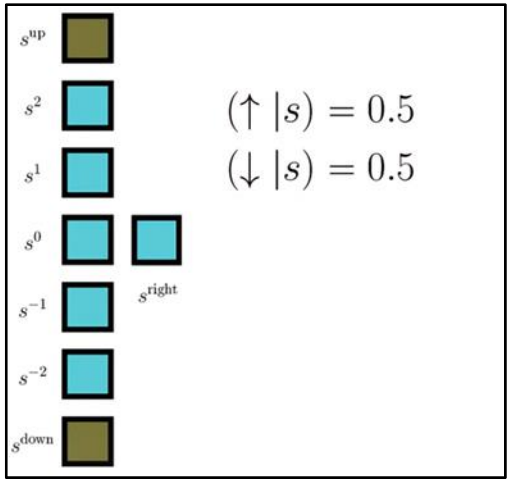
- Trajectories have to terminate
- Exploration-Exploitation *dichotomy*:
  - To discover optimal policies, we must **explore** all state-action pairs.
  - To get high returns we must **exploit** known high state-action pairs.

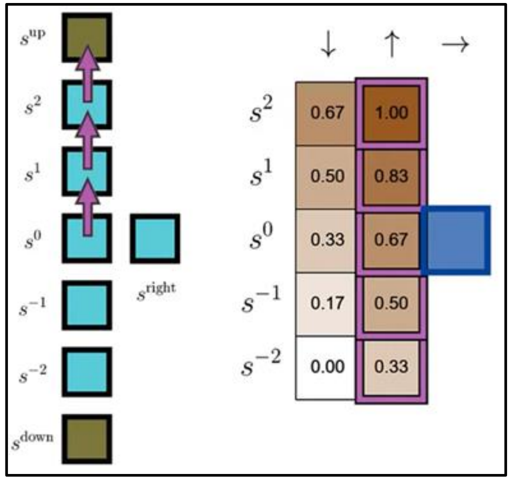




Caveats of MC

- 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, π_* is always discoverable if the policy is **Soft**:

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

ϵ-greedy policy of Q: With probability ϵ, take an action selected uniformly from $\mathcal{A}(s)$, otherwise take $\mathrm{argmax}_a\, Q(s,a)$.

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-α MC for estimating $\pi \approx \pi*$

Algorithm inputs:

ϵ α M

Initialize arbitrarily:

$\pi \leftarrow$ some ϵ-soft policy

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

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

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

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

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

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

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

Off-Policy Constant-α 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 $m = 1, \cdots, M$:

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

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

$$\boxed{\begin{aligned} &\rho_t^m \leftarrow \prod_{\tau=t+1}^{T_m-1} \frac{\pi(a_\tau^m|s_\tau^m)}{b(a_\tau^m|s_\tau^m)} \qquad \text{(or 1 if } t+1 > T_m - 1\text{)} \\ &g_t^m \leftarrow \rho_t^m (r_{t+1}^m + \gamma r_{t+2}^m + \cdots) \end{aligned}}$$

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

$\pi(s_t^m) \leftarrow \text{argmax}_a\, Q(s_t^m, a)$ (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:*

$$q_\pi(s,a) = E_\pi[G_t|(S_t,A_t)=(s,a)] = E_\pi[R_{t+1}+\gamma q_\pi(S_{t+1},A_{t+1})|(S_t,A_t)=(s,a)]$$

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

- MC approach:

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

$$Q(s_t^m,a_t^m) \leftarrow Q(s_t^m,a_t^m) + \alpha\big(\underbrace{g_t^m}_{\text{target}} - Q(s_t^m,a_t^m)\big)$$

Temporal Difference Learning

- *a priori:*

$$q_\pi(s,a) = E_\pi[G_t|(S_t,A_t)=(s,a)] = E_\pi[R_{t+1}+\gamma q_\pi(S_{t+1},A_{t+1})|(S_t,A_t)=(s,a)]$$

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

- MC approach:

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

target

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

- 1-step TD approach:

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

bootstrap

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

target

SARSA

Constant-α MC for estimating $\pi \approx \pi*$

Algorithm inputs:

ϵ α M

Initialize arbitrarily:

$\pi \leftarrow$ some ϵ-soft policy

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

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

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

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

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

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

$\pi \leftarrow \epsilon\text{-greedy}(Q)$

On Policy TD Control: n-step SARSA

Changes:

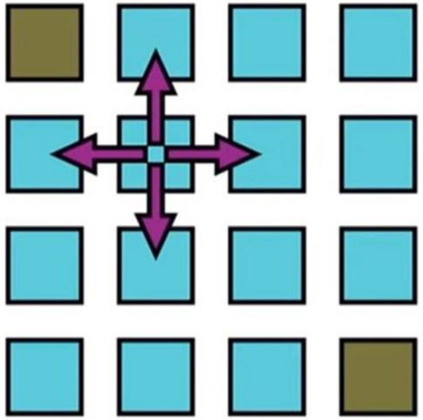
- Approximating the rewards beyond the n-th step with the current value of $Q(s,a)$ (bootstrapping):

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

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

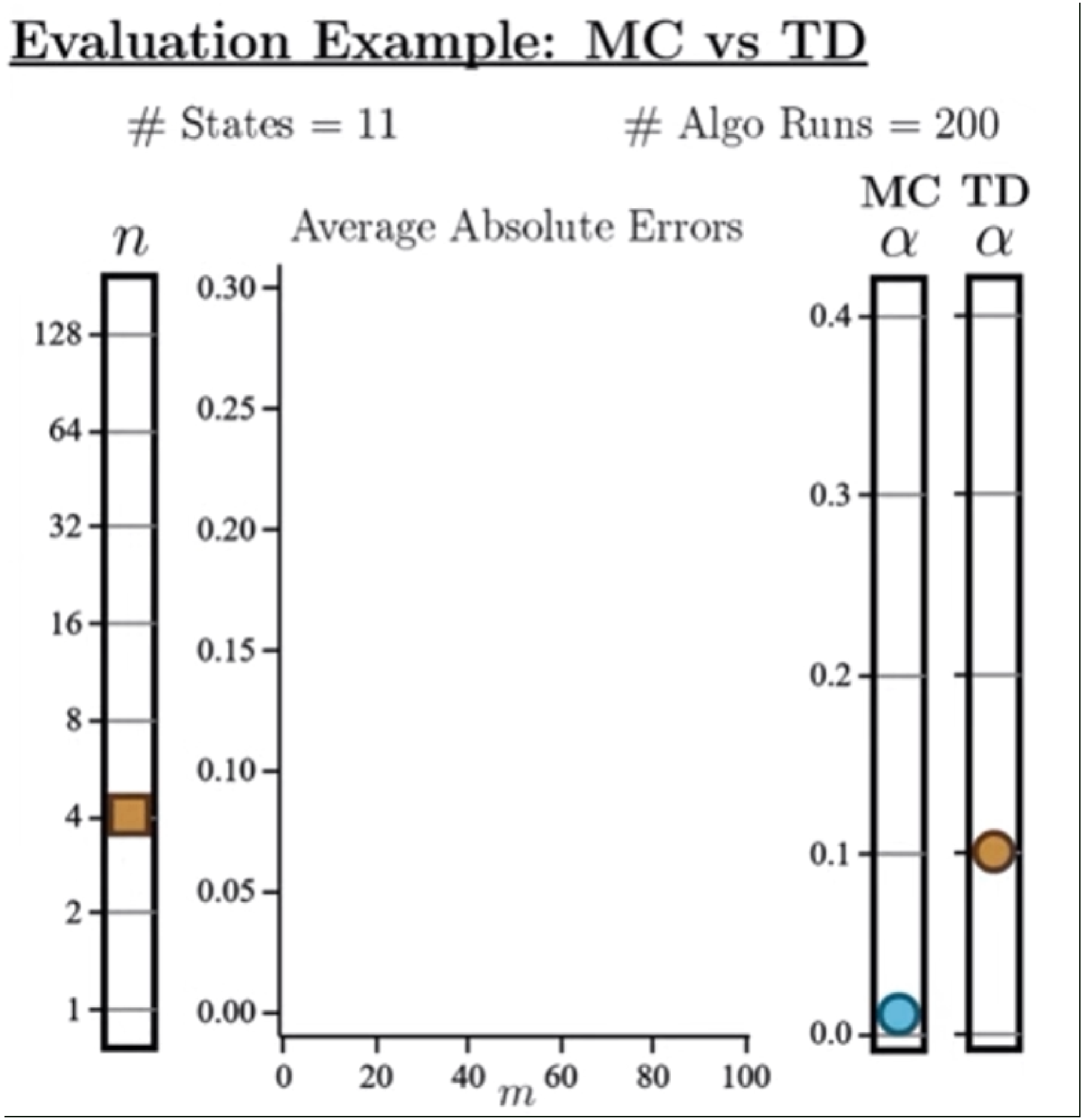
- Updates happen *during* the episode, Interweaving between (S, A, R) tuples, with an n step delay.
- The policy is updated in similar manner with MC

TD ∋ MC

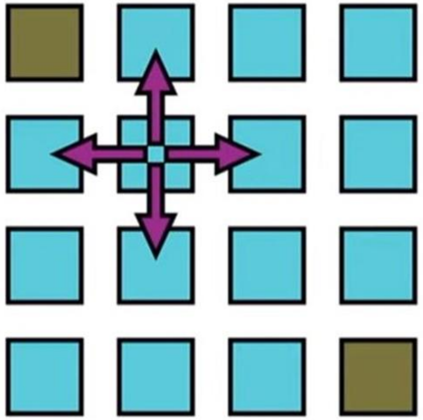


$R_t = -1$

$\pi(a|s) = 0.25$

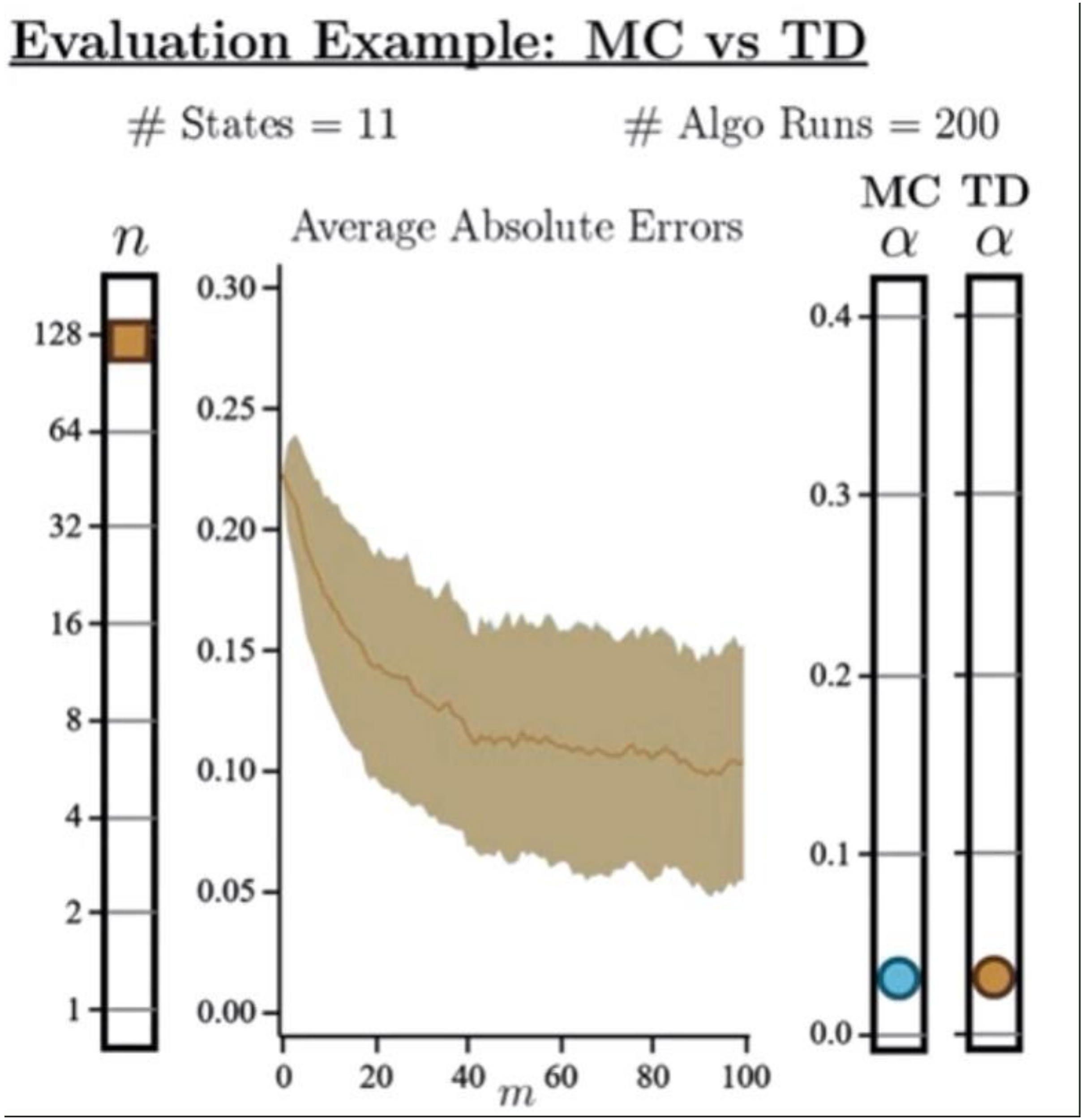


TD ∋ MC



$R_t = -1$

$\pi(a|s) = 0.25$



Q-learning

Constant-α MC for estimating $\pi \approx \pi*$

Algorithm inputs:

ϵ α M

Initialize arbitrarily:

$\pi \leftarrow$ some ϵ-soft policy

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

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

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

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

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

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

$\pi \leftarrow \epsilon\text{-greedy}(Q)$

Q-Learning

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

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

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

$$s_0^m, a_0^m, r_1^m, s_1^m, a_1^m, r_2^m, s_2^m, a_2^m, r_3^m, s_3^m, a_3^m, r_4^m \cdots$$

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 → never enough data.
- Example:
  - Assume $v_\pi(s) = \hat{v}(s, w)$, $\hat{v}(s+\delta, w) = \hat{v}(s+\delta, w) + \left(\frac{\partial \hat{v}}{\partial s}\right)^\top \delta$.
  - Goal:

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

  - 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 → w is param of DNN.

Function Approximation

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

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

  - Update rule:

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

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

Function Approximation

- What about the policy function?

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 α

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

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

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

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

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

- $\nabla_\theta \ln\pi(a_t|s_t, \theta)$ gives the "direction" that increasing θ will increase $\pi(a_t|s_t, \theta)$.

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

Policy Gradient Methods

REINFORCE

To specify upfront:

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

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

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

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

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

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

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

Policy Gradient Methods

REINFORCE

To specify upfront:

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

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

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

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

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

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

- $\nabla_\theta \ln\pi(a_t|s_t,\theta)$ gives the "direction" that increasing θ will increase $\pi(a_t|s_t,\theta)$.
- The increase of θ 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

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

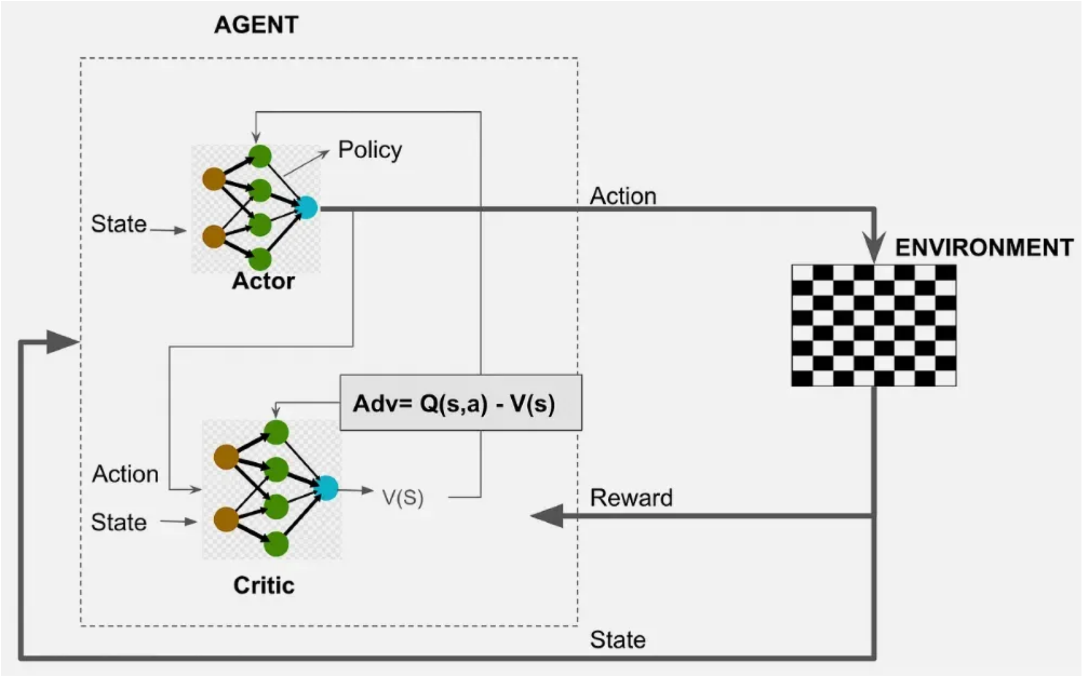
→ provides a measure of how "good" and action is compared with the average action.

- "Actor" gradient:

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

- "Critic" gradient:

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

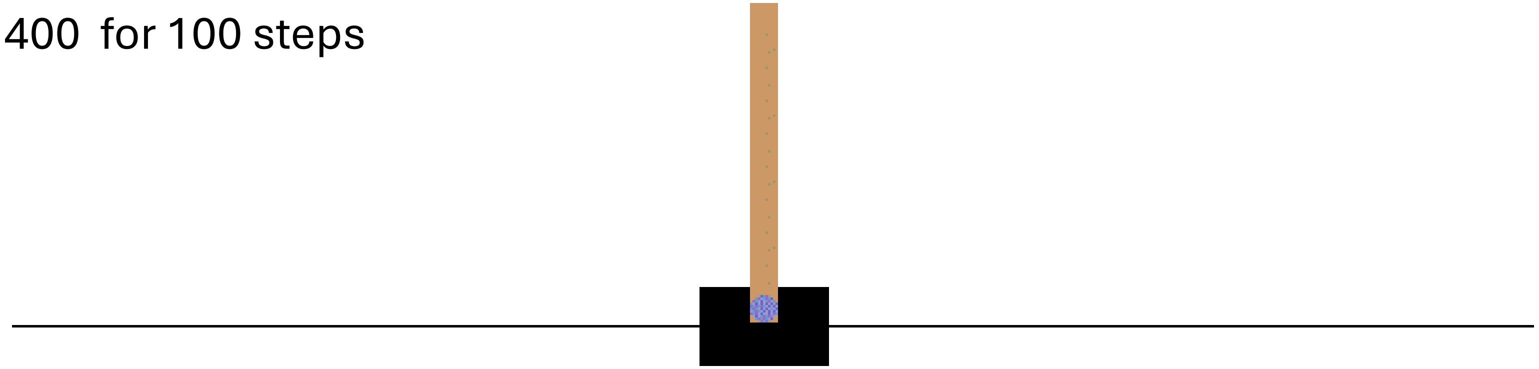


Tutorial: CartPole by REINFORCE method

https://www.gymlibrary.dev/environments/classic_control/cart_pole/

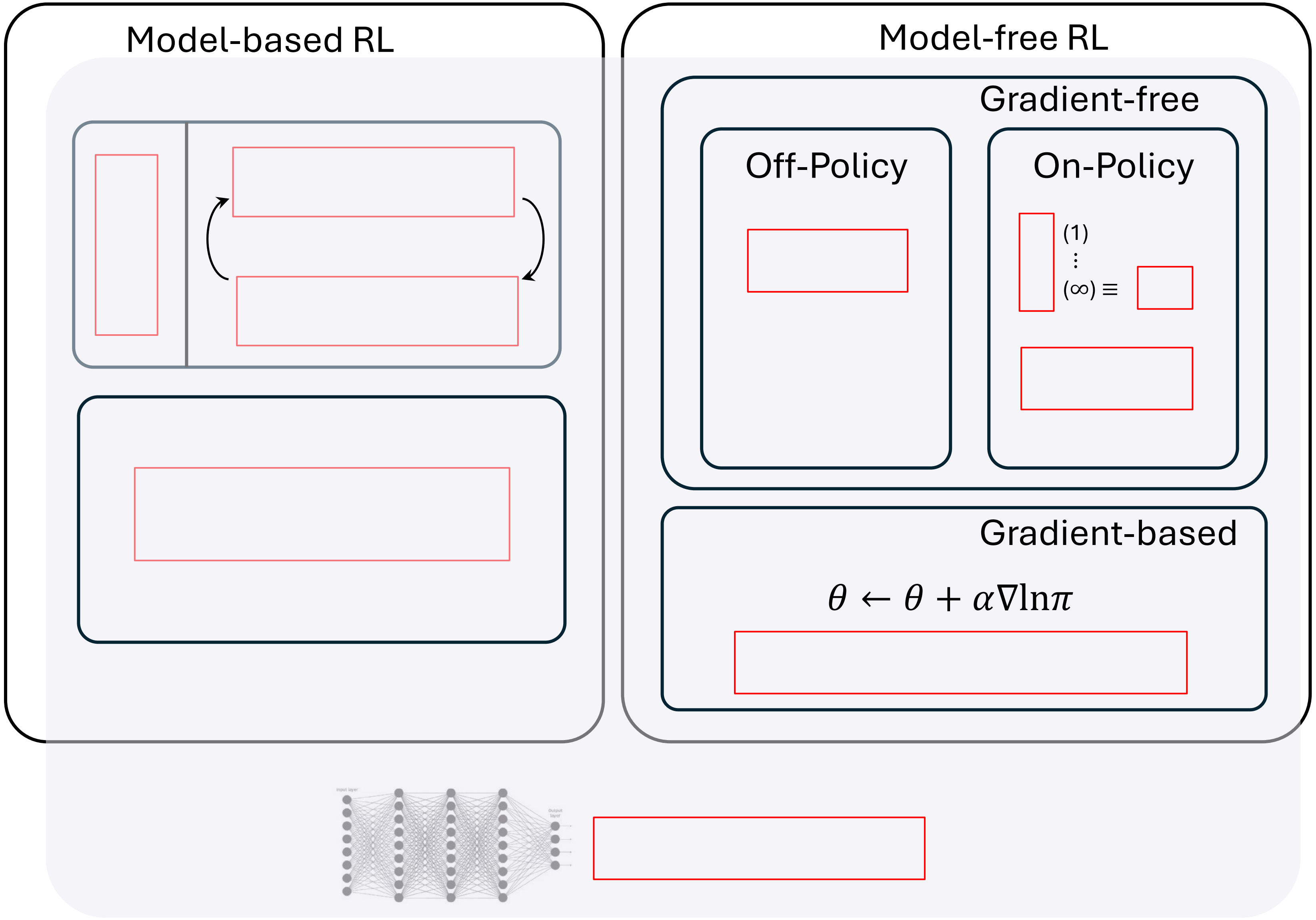
https://tinyurl.com/tmn2025DRLCartpole

- End when score = 050 for 100 steps
- End when score = 100 for 100 steps
- End when score = 200 for 100 steps
- End when score = 400 for 100 steps



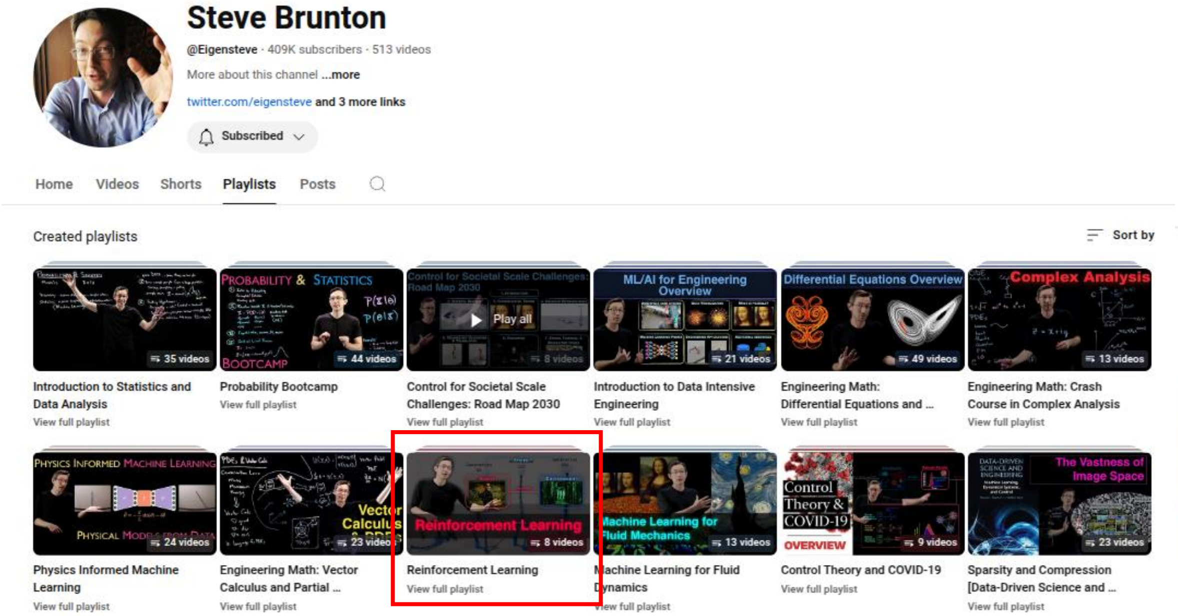
Summary

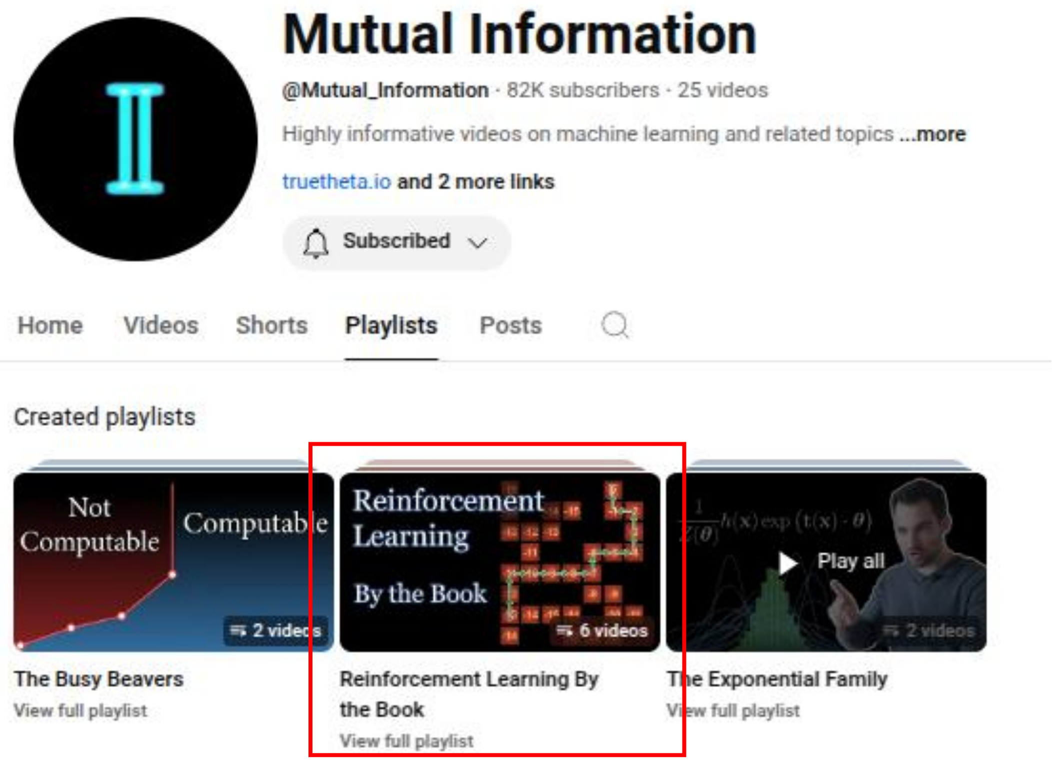
- Value Evaluation, Policy Iteration, Policy Improvement, Value Iteration…
- MC
- TD, Q-learning, Sarsa
- Function Approximation (Deep RL)
- Policy Gradient Methods



References

- Reinforcement Learning: An Introduction, Sutton
- David Silver RL Lectures
- Zhao Shiyu RL Lectures
- OpenAI Introduction to RL






DRL Bootcamp