Robust Adversarial Reinforcement Learning Explained

Shuijing Liu
3/11/2019

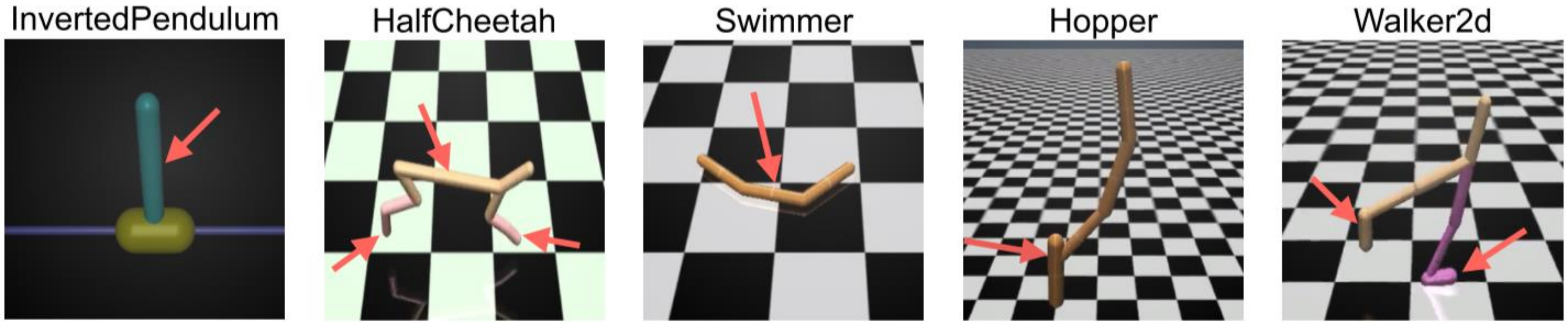
Overview of RARL

Motivation

- Challenges in deep RL for real-world policy learning:
  - Due to scarcity of data, training is often restricted to a limited set of scenarios, which causes overfitting
  - If we learn a policy in simulator and transfer it to the real world, the gap between simulator and the real world may cause unsuccessful transfer, if the policy is not robust enough
- Training more robust policies using less data:
  - The gap between simulations and real-world can be viewed as external forces/disturbances in the system
  - The adversary disturbance can be learned and reinforced to impede the agent from achieving its goal

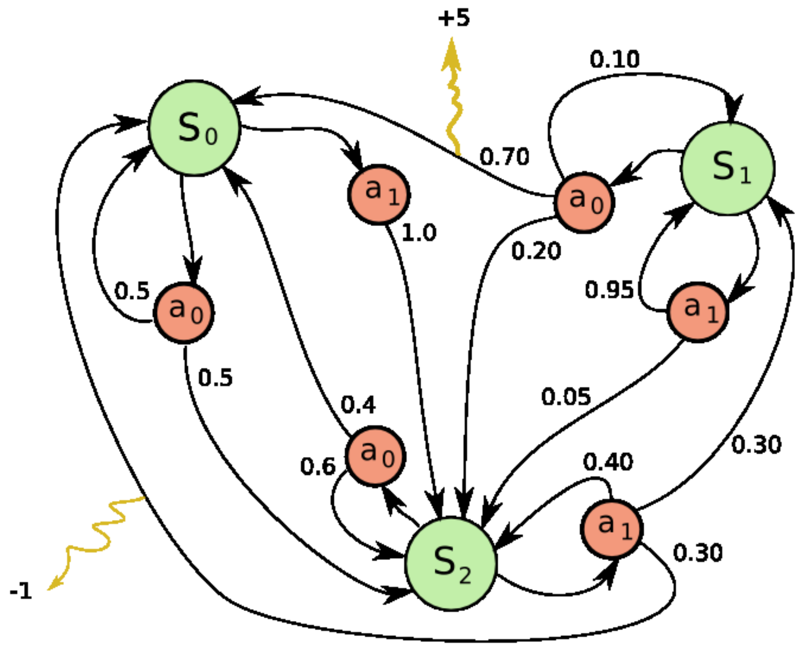
Adversary disturbance examples

https://gym.openai.com/envs/#mujoco



Background

Markov decision process



- A Markov decision process (MDP) consists of:
  - $S = \{s_1, \dots, s_n\}$: a finite set of states
  - $A = \{a_1, \dots, a_m\}$: a finite set of actions
  - $P(s'|s, a)$: the probability that if the agent takes action a in state s at time t, it will end up in state s' at time $(t + 1)$
  - $R(s, a)$: the immediate reward received after taking action a at state s

Reinforcement Learning

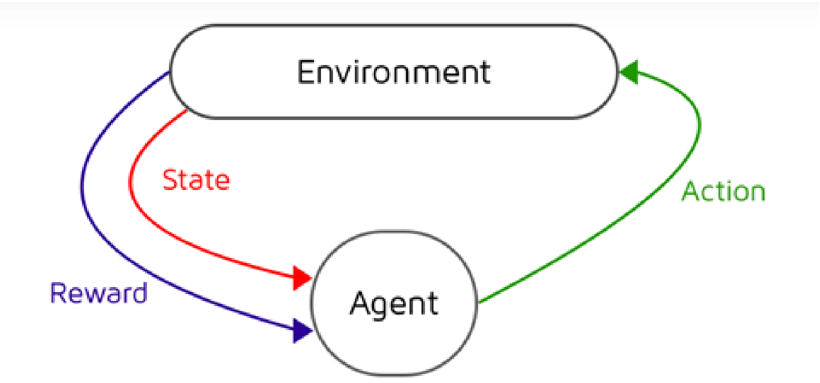
- The agent doesn't know transition probability or reward function
- The agent's action selection is called *policy* π:

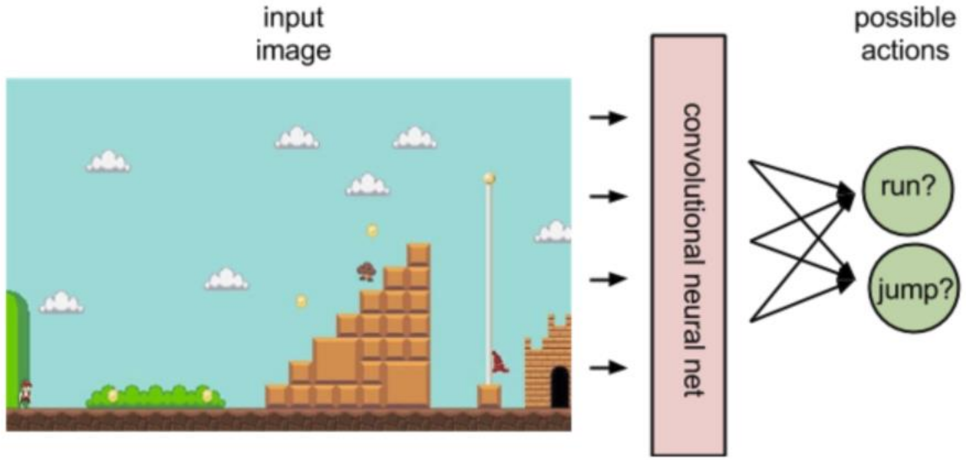
$$\begin{cases} non-probabilistic\ policy: a_t \coloneqq \pi(s_t) \\ probabilistic\ policy: \pi(a|s) = p(a_t = a|s_t = s) \end{cases}$$

- Value function $V_\pi(s)$ is defined as the expected return starting with state s and following policy π:

$$V_\pi(s) = E[\sum_{t=0}^{\infty} \gamma^t r(s,a)|s_0 = s]$$

- We want to find a policy with maximum expected long-term reward $R = \sum_{t=0}^{\infty} \gamma^t r_t$, where $\gamma \in [0,1]$ is the discount rate
- When the state space or state dimensionality is large, Deep RL can be used to approximate $\pi(s)$



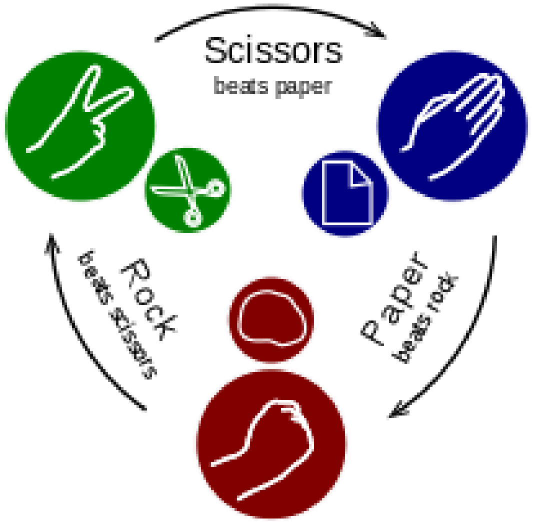


2-player zero-sum games

- In a two-player zero-sum game, player A's gain is exactly balanced by player B's loss, and player A's loss is exactly balanced by player B's gain.
- The MDP of two-player game can be expressed as a tuple $(S, A_1, A_2, P, r, \gamma, s_0)$, where
  - A_1 and A_2 are the sets of actions player 1(protagonist) and player 2(antagonist) can take
  - $P: S \times A_1 \times A_2 \rightarrow R$ is the transition probability
  - $r: S \times A_1 \times A_2$ is the reward function for both players
- If protagonist is playing strategy μ and antagonist is playing strategy v, the reward function is

$$r_{\mu,v} = E_{a^1 \sim \mu(s), a^2 \sim v(s)}[r(s, a^1, a^2)]$$

- A zero-sum two-player game can be seen as protagonist maximizing the long term γ discounted reward while antagonist is minimizing it.



Rock, Paper, Scissors	Rock	Paper	Scissors
Rock	0, 0	-1, 1	1, -1
Paper	1, -1	0, 0	-1, 1
Scissors	-1, 1	1, -1	0, 0

RARL Formulation

Problem Formulation of RARL

- At every time step t, both players observe the state s_t and take actions $a_t^1 \sim \mu(s_t)$ and $a_t^2 \sim v(s_t)$, respectively
- The state transition $s_{t+1} = P(s_t, a_t^1, a_t^2)$ and reward $r_t = r(s_t, a_t^1, a_t^2)$ is observed from the environment
- In our zero-sum game, the protagonist gets a reward $r_t^1 = r_t$ while the adversary get a reward $r_t^2 = -r_t$
- Therefore, the MDP can be represented as $(s_t, a_t^1, a_t^2, r_t^1, r_t^2, s_{t+1})$
- The protagonist tries to maximize the reward function

$$R^1 = E_{s_0 \sim \rho, a^1 \sim \mu(s), a^2 \sim v(s)} [\sum_{t=0}^{T-1} r(s, a^1, a^2)]$$

- The Nash equilibrium for this game is $R^{1*} = \min_{v} \max_{\mu} R^1(\mu, v) = \max_{\mu} \min_{v} R^1(\mu, v)$ (i.e. minimizing one's own maximum loss, and maximizing one's own minimum gain)

The Algorithm

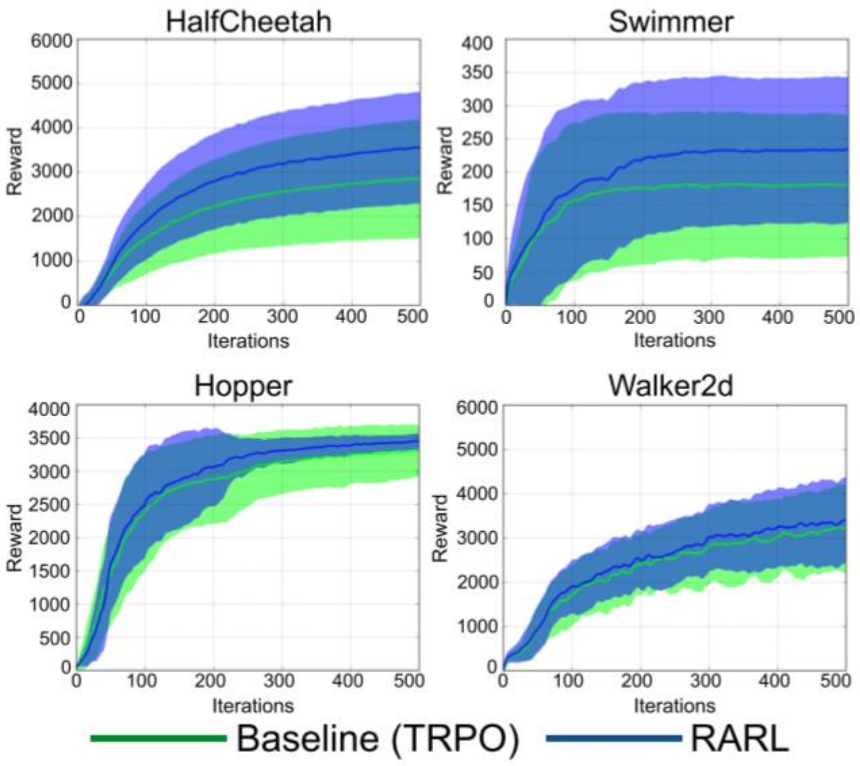
- RARL algorithm optimizes **both agents' policies** alternatively:
  - In the first phase, protagonist learns a policy while holding the adversary's policy fixed
  - Next, the protagonist's policy is fixed and the adversary's policy is learned
  - Repeat these two phases until convergence

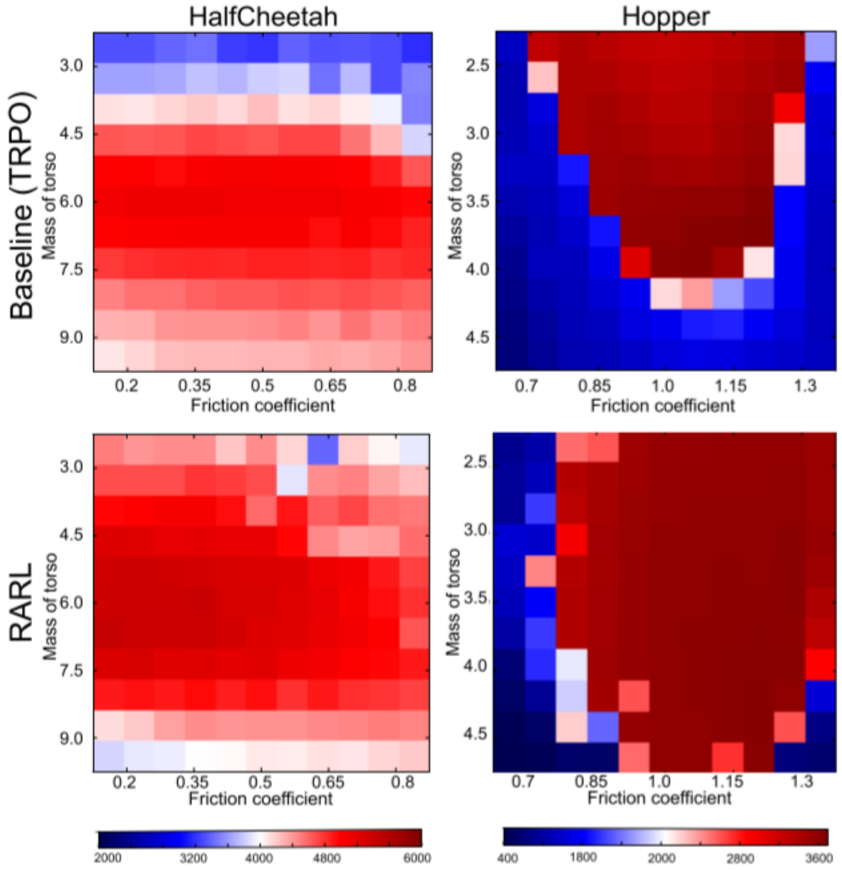
Algorithm 1 RARL (proposed algorithm)

Input: Environment $\mathcal{E}$; Stochastic policies μ and ν
Initialize: Learnable parameters θ_0^μ for μ and θ_0^ν for ν
for i=1,2,..N_{iter} **do**
  $\theta_i^\mu \leftarrow \theta_{i-1}^\mu$
  for j=1,2,..N_μ **do**
    $\{(s_t^i, a_t^{1i}, a_t^{2i}, r_t^{1i}, r_t^{2i})\} \leftarrow \text{roll}(\mathcal{E}, \mu_{\theta_i^\mu}, \nu_{\theta_{i-1}^\nu}, N_{\text{traj}})$
    $\theta_i^\mu \leftarrow \text{policyOptimizer}(\{(s_t^i, a_t^{1i}, r_t^{1i})\}, \mu, \theta_i^\mu)$
  end for
  $\theta_i^\nu \leftarrow \theta_{i-1}^\nu$
  for j=1,2,..N_ν **do**
    $\{(s_t^i, a_t^{1i}, a_t^{2i}, r_t^{1i}, r_t^{2i})\} \leftarrow \text{roll}(\mathcal{E}, \mu_{\theta_i^\mu}, \nu_{\theta_i^\nu}, N_{\text{traj}})$
    $\theta_i^\nu \leftarrow \text{policyOptimizer}(\{(s_t^i, a_t^{2i}, r_t^{2i})\}, \nu, \theta_i^\nu)$
  end for
end for
Return: $\theta_{N_{\text{iter}}}^\mu, \theta_{N_{\text{iter}}}^\nu$

Evaluation and Results

The robustness of RARL policies were compared against baseline policies:





Video demonstrations

- https://www.youtube.com/watch?v=esxUd4tP2G8
- Some results from my previous work