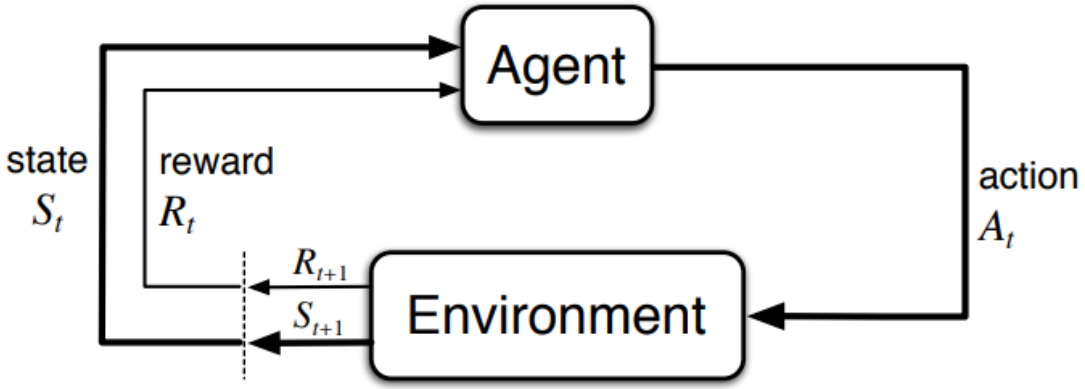
Deep Reinforcement Learning (II)

Policy Gradient Methods

SUN Yinghan

2022. 03. 15

Recap: Markov Decision Process



	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

Markov property: Current state completely characterizes the state of the world.

A Markov decision process is defined by a tuple: $< \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{P}, \gamma >$.

- $\mathcal{S}$: set of possible states — State = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}
- $\mathcal{A}$: set of possible actions — Action = {up, down, left, right}
- $\mathcal{R}$: distribution of reward — Reward = -1 for all transitions
- $\mathbb{P}$: transition probability — $p(s', r|s, a) = \Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\}$
- γ: discount factor

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots \right) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

Recap: Value-Based Approach

Policy: ways of acting. It is a mapping from states to probabilities of selecting each possible action.

Value Function: the expected return when starting in s and following π thereafter.

State-value function for policy π: $v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$
Action-value function for policy π: $q_\pi(s,a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$

Relationship:

$$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s,a)$$

$$q_\pi(s,a) = \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma v_\pi(s')]$$

Objective: find the optimal policy, which maximizes cumulative discounted reward (value function).

Q-Learning: a temporal-difference learning method.

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R + \gamma \max_a Q(s',a) - Q(s,a)]$$

What if there are large number of states, or even infinitely many states? **DQN**

Policy-Based Methods

RL Goal: Learning a policy to maximize the expectation of the accumulate returns (rewards) when an agent interacts with the environment.

$$\theta^* = \arg\max_\theta \underbrace{\mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_t r(s_t, a_t) \right]}_{J(\theta)}$$

Value-Based Methods: learning action-value firstly, then select actions based on the learned action-value (policy).

Policy-Based Methods: compute the gradient of J(θ) w.r.t. θ directly, then update the parameter θ of the policy according to the gradient.

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_t r(s_t, a_t) \right] = \int p_\theta(\tau) r(\tau) \mathrm{d}\tau$$

$$\begin{aligned}
\nabla_\theta J(\theta) &= \int \nabla_\theta p_\theta(\tau) r(\tau) \mathrm{d}\tau \\
&= \int p_\theta(\tau) \frac{\nabla_\theta p_\theta(\tau)}{p_\theta(\tau)} r(\tau) \mathrm{d}\tau \\
&= \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) r(\tau) \mathrm{d}\tau \\
&= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\nabla_\theta \log p_\theta(\tau) r(\tau) \right]
\end{aligned}$$

Compute the Gradient

Question: How to compute $\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\nabla_\theta \log p_\theta(\tau) r(\tau) \right]$?

$$
\begin{aligned}
p_\theta(\tau) &= p_\theta(s_1, a_1, s_2, a_2, s_3, a_3, \cdots, s_T, a_T) \\
&= p(s_1) p(a_1|s_1) p(s_2|s_1, a_1) p(a_2|s_2) p(s_3|s_1, a_1, s_2, a_2) p(a_3|s_3) \cdots \\
&= p(s_1) \pi(a_1|s_1) p(s_2|s_1, a_1) \pi(a_2|s_2) p(s_3|s_2, a_2) \pi(a_3|s_3) \cdots \\
&= p(s_1) \prod_{t=1}^{T} \left[\pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t) \right]
\end{aligned}
$$

$$
\log p_\theta(\tau) = \log p(s_1) + \sum_{t=1}^{T} \left[\log \pi_\theta(a_t|s_t) + \log p(s_{t+1}|s_t, a_t) \right]
$$

$$
\begin{aligned}
\nabla_\theta \log p_\theta(\tau) &= \nabla_\theta \log p(s_1) + \sum_{t=1}^{T} \left[\nabla_\theta \log \pi_\theta(a_t|s_t) + \nabla_\theta \log p(s_{t+1}|s_t, a_t) \right] \\
&= \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)
\end{aligned}
$$

$$
\Rightarrow \nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t=1}^{T} r(s_t, a_t) \right]
$$

Monte-Carlo Sampling: $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left(\sum_{t=1}^{T} r(s_{i,t}|a_{i,t}) \right) \right]$

Algorithm: REINFORCE

1. sample $\{\tau^i\}$ from $\pi_\theta(a_t|s_t)$. (i.e. run the policy)
2. compute gradient of the cost function.

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left[\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\right)\left(\sum_{t=1}^{T} r(s_{i,t}|a_{i,t})\right)\right]$$

3. update policy parameters: $\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta)$

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left[\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\right)\left(\sum_{t=1}^{T} r(s_{i,t}|a_{i,t})\right)\right]$$

We need to choose some proper policies with parameter θ, or use a neural network to approximate this mapping.

We need to design some proper reward functions.

Example: Gaussian Policies (Continuous Case)

1. sample $\{\tau^i\}$ from $\pi_\theta(a_t|s_t)$. (i.e. run the policy)
2. compute gradient of the cost function.

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left[\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\right)\left(\sum_{t=1}^{T} r(s_{i,t}|a_{i,t})\right)\right]$$

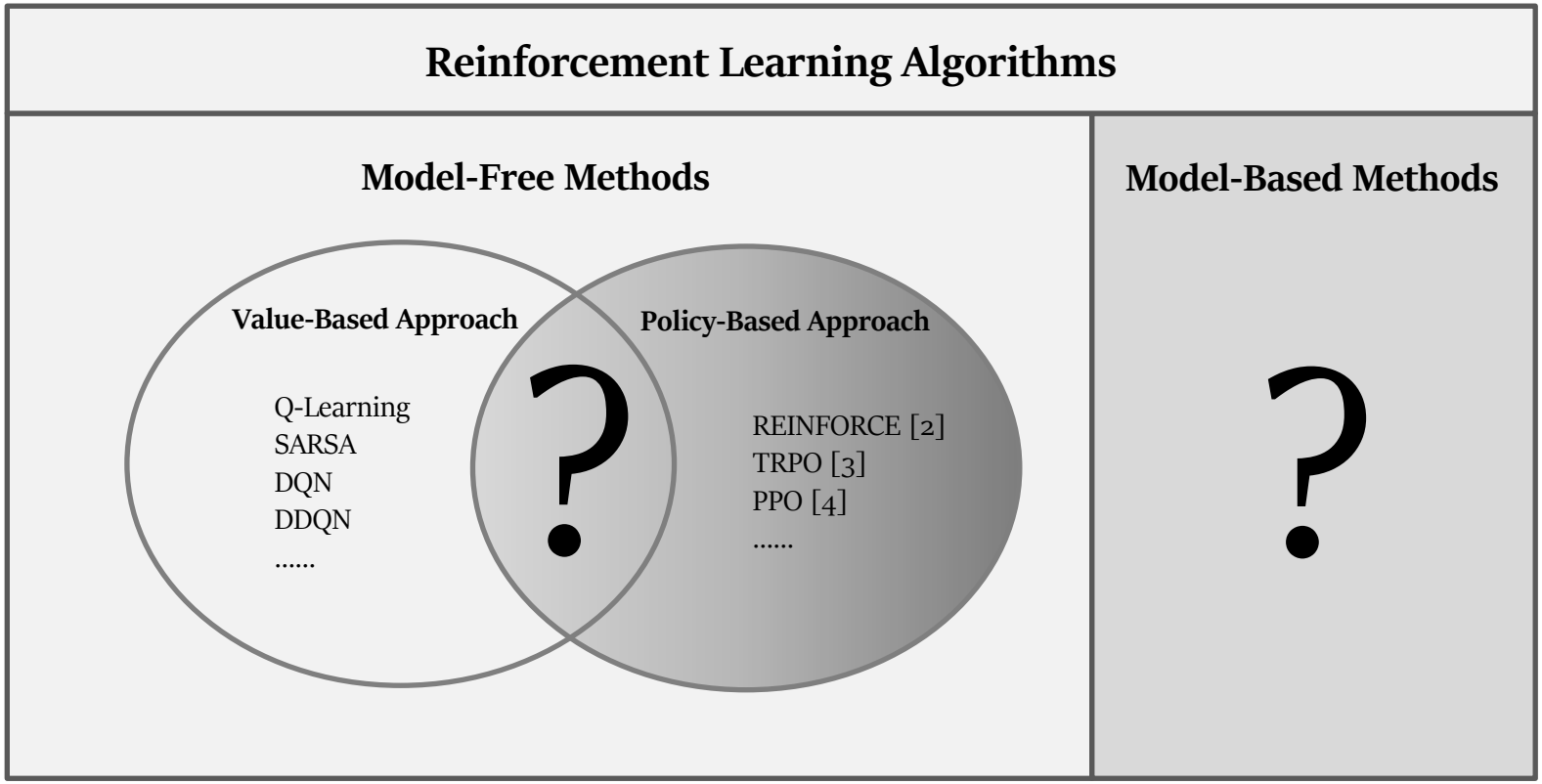
3. update policy parameters: $\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta)$

$$\begin{aligned}\pi_\theta(a_t|s_t) &= \mathcal{N}(g_\theta(s_t), \sigma^2)\\ &= \frac{1}{\sigma\sqrt{2\pi}}\exp\left\{-\frac{1}{2}\frac{(a_t - g_\theta(s_t))^2}{\sigma^2}\right\}\end{aligned}$$

$$\log \pi_\theta(a_t|s_t) = -\frac{1}{2\sigma^2}(a_t - g_\theta(s_t))^2 + \log\frac{1}{\sqrt{2\pi}\sigma}$$

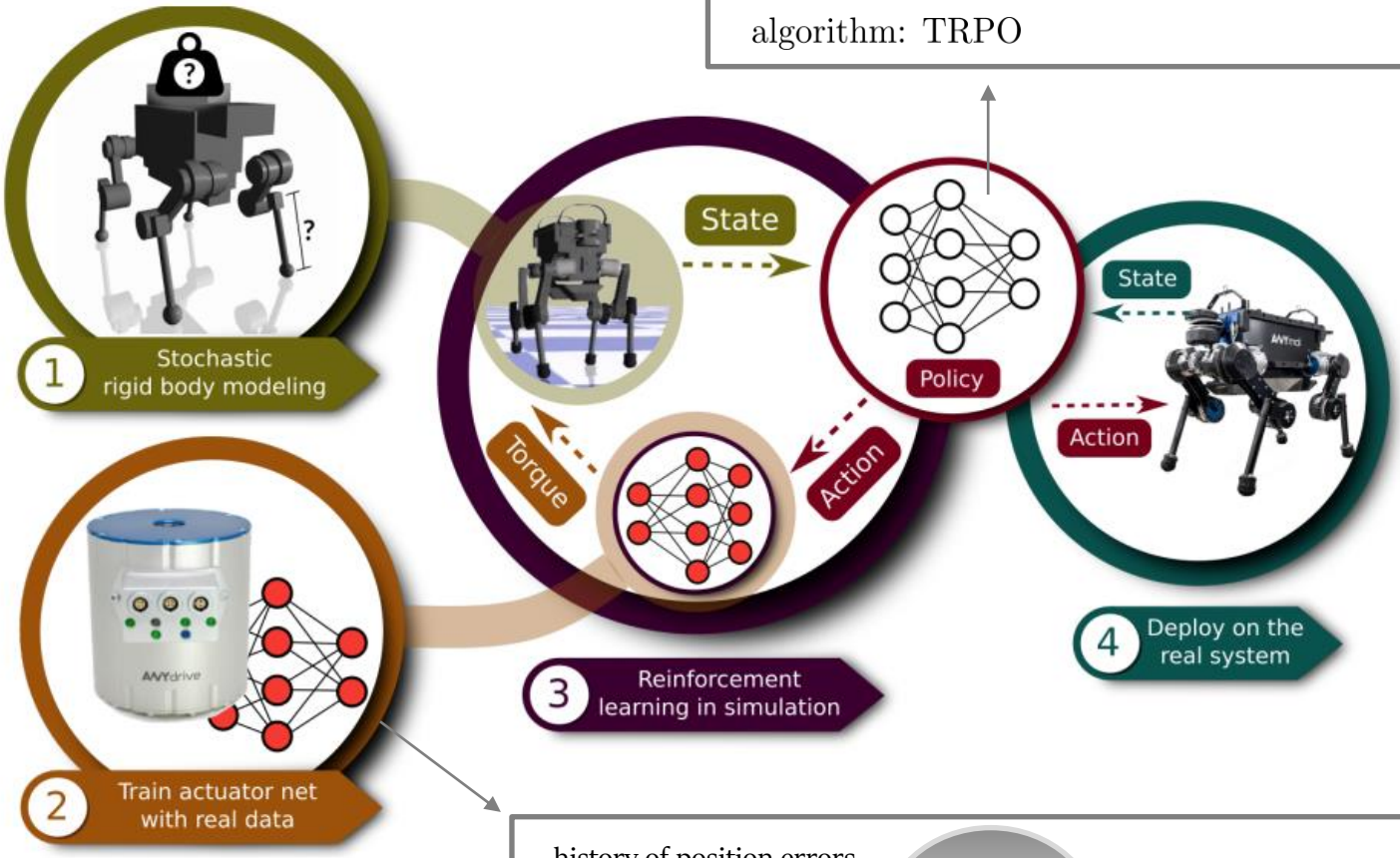
$$\nabla_\theta \log \pi_\theta(a_t|s_t) = \frac{1}{\sigma^2}(a_t - g_\theta(s_t))\frac{\partial g_\theta(s_t)}{\partial \theta}$$

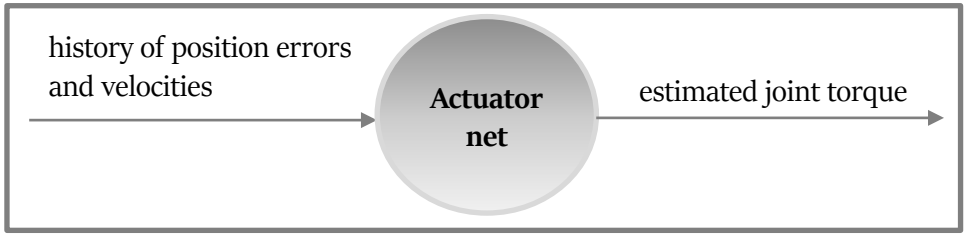
Summary: RL Algorithms (So Far)



Case Study: Application on Robot Locomotion

state (observation): $\{\theta, h_b, v_b, \omega_b, q, \dot{q}, q_h, a_{k-1}, C\}$
action: low-impedance joint position commands
algorithm: TRPO

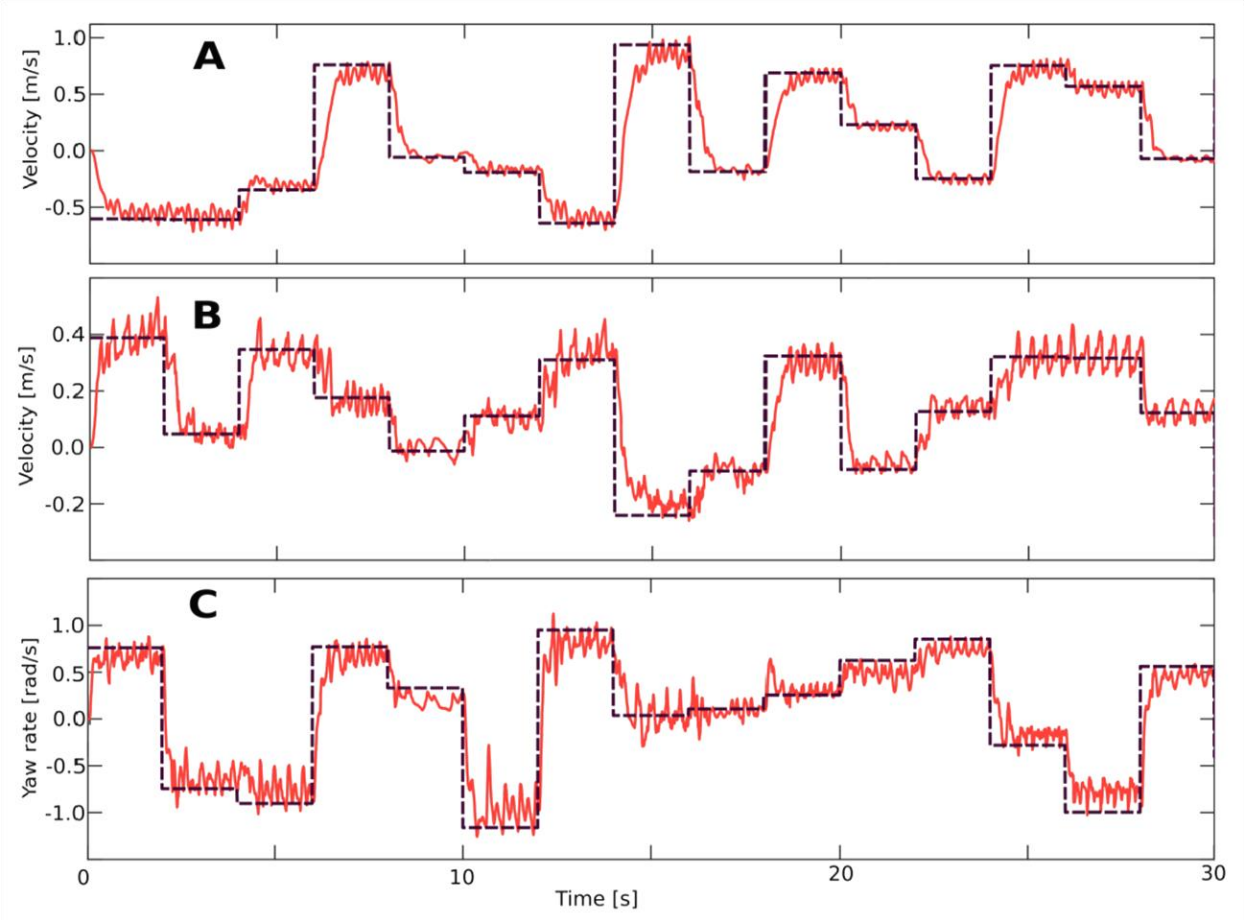




Hwangbo J, Lee J, Dosovitskiy A, et al. Learning Agile and Dynamic Motor Skills for Legged Robots[J]. Science Robotics, 2019, 4(26): eaau5872.

Case Study: Application on Robot Locomotion

Base Velocity Tracking Performance of the Learned Controller While Following Random Commands



(A) Forward velocity, (B) Lateral velocity, (C) yaw rate. For all graphs, the dotted lines represent the commanded velocity and the solid lines represent the measured velocity. All commands are followed with a reasonable accuracy even when the commands are given in a random fashion.

Hwangbo J, Lee J, Dosovitskiy A, et al. Learning Agile and Dynamic Motor Skills for Legged Robots[J]. Science Robotics, 2019, 4(26): eaau5872.

References

[1] Sutton R S, Barto A G. Reinforcement Learning: An Introduction[M]. MIT press, 2018.

[2] Williams R J. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning[J]. Machine learning, 1992, 8(3): 229-256.

[3] Schulman J, Levine S, Abbeel P, et al. Trust Region Policy Optimization[C]//International Conference on Machine Learning. PMLR, 2015: 1889-1897.

[4] Schulman J, Wolski F, Dhariwal P, et al. Proximal Policy Optimization Algorithms[J]. arXiv preprint arXiv:1707.06347, 2017.

[5] Hwangbo J, Lee J, Dosovitskiy A, et al. Learning Agile and Dynamic Motor Skills for Legged Robots[J]. Science Robotics, 2019, 4(26): eaau5872.

Thanks for Listening