DDPG and Soft AC

Spring, 2022

Outline

1. Deep Deterministic Policy Gradient
2. Soft Actor-Critic

Table of Contents

Q-learning-DDPG-TD3-SAC

1. Value based RL methods starting from Q-learning are also being developed over the years
2. **DDPG**: Deterministic Policy Gradient Algorithms, Silver et al. ICML 2014
3. **TD3**: Addressing Function Approximation Error in Actor-Critic Methods, Fujimoto et al. ICML 2018
4. **SAC**: Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, Haarnoja et al. ICML 2018

Stochastic Policy

- The policy is stochastic and denoted by

$$\pi_\theta : \mathcal{S} \to \mathcal{P}(\mathcal{A}),$$

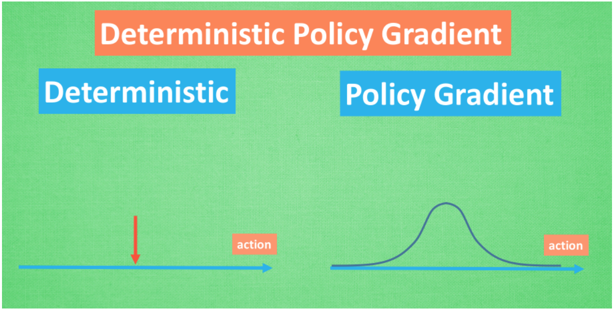
  where $\mathcal{P}(\mathcal{A})$ is the set of probability measures on $\mathcal{A}$ and $\theta \in \mathbb{R}^n$ is a vector of n parameters.
- $\pi_\theta(a_t|s_t)$ is the conditional probability density at a_t associated with the policy.
- For the same state s, the actions stochastically selected according to θ might be different.

Deterministic Policy

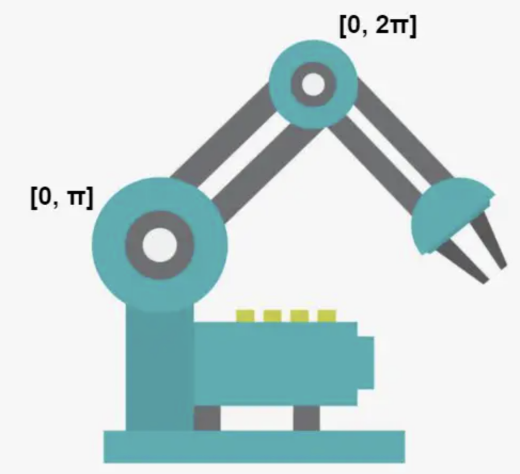
- Select a deterministic action:

$$a = \mu_\theta(s)$$

- action is *uniquely* determined at the state s w.r.t the parameter θ
- suitable for the continuous action space, especially if the action space has many dimensions

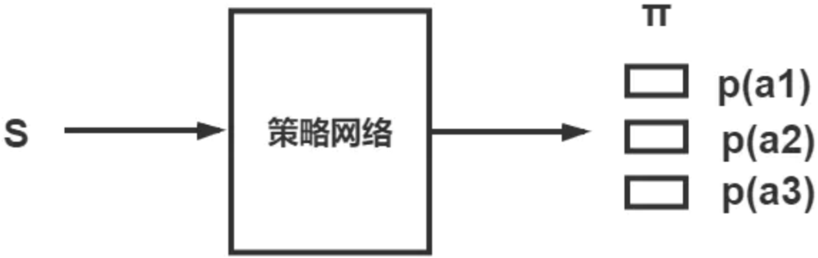


Continuous Action Space

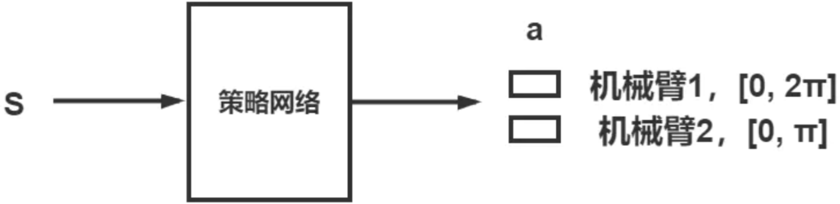


$$A \in [0, 2\pi] * [0, \pi]$$

DQN for Continuous Action Space



Deterministic Policy Gradient



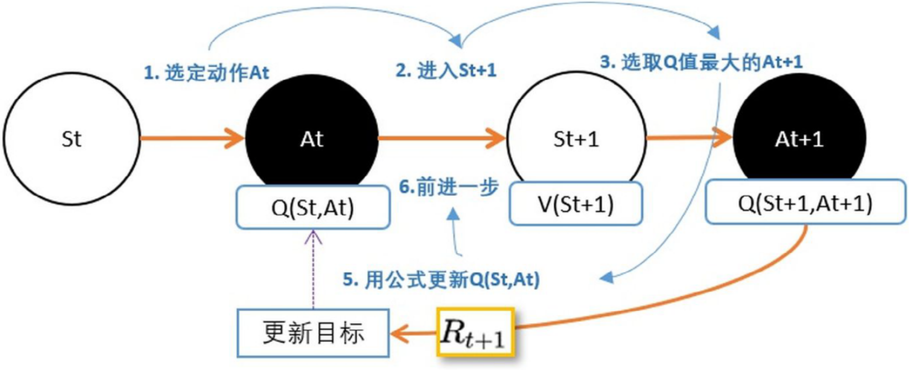
Deep Deterministic Policy Gradient (DDPG)

1. Motivation: how to extend DQN to the environment with continuous action space?
2. DDPG is very similar to DQN, which can be considered as a continuous action version of DQN

$$\textbf{DQN}: a^* = \arg\max_a Q^*(s,a)$$
$$\textbf{DDPG}: a^* = \arg\max_a Q^*(s,a) \approx Q_\phi(s, \mu_\theta(s))$$

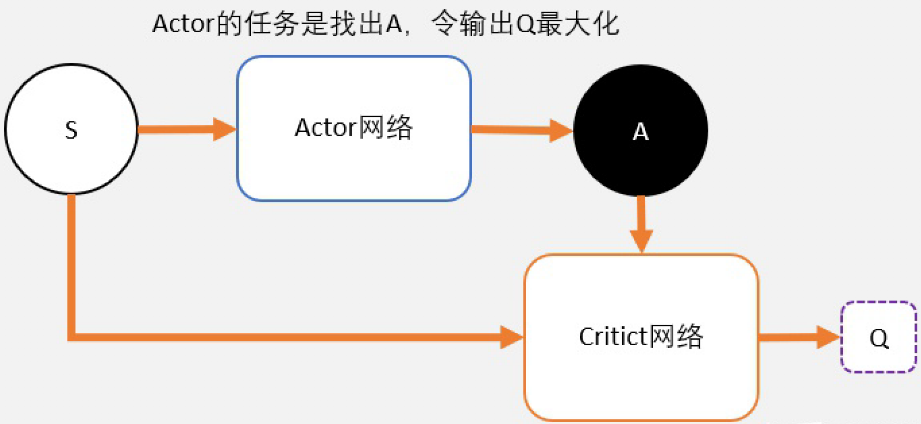
   1. a deterministic policy $\mu_\theta(s)$ directly gives the action that maximizes $Q_\phi(s,a)$
   2. as action a is continious we assume Q-function $Q_\phi(s,a)$ is differentiable with respect to a

Q Function of DQN



$$Q(S, A) \leftarrow Q(S, A) + \alpha\left[R + \gamma \max_a Q\left(S', a\right) - Q(S, A)\right]$$

Actor Network of DDPG



Actor Network of DDPG (2)

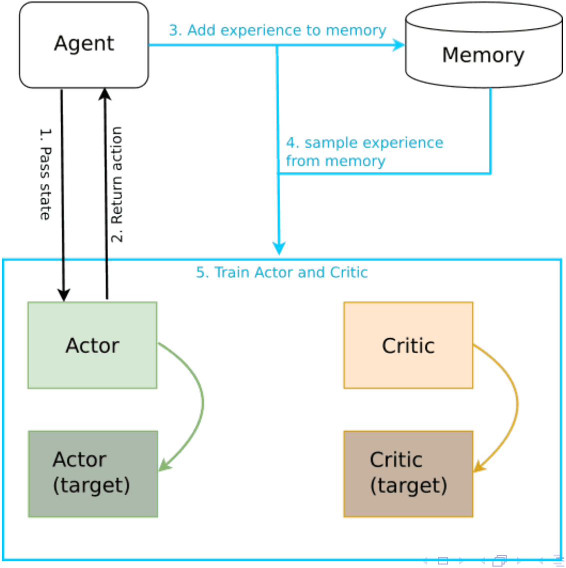
Critic 输出的价值代表了Actor预测动作的好坏，因此策略网络的目标是最大化价值 $Value$ ，自然就想到了用梯度上升法来最大化 $q(s,a;w)$,于是，我们可以对 $q(s,a;w)$ 求 θ 的梯度，让我们将策略网络记作 $\pi(s;\theta)$ ：

$$pg = \frac{\partial q(s,\pi(s;\theta);w)}{\partial \theta} = \frac{\partial q(s,a;w)}{\partial a} * \frac{\partial a}{\partial \theta}$$

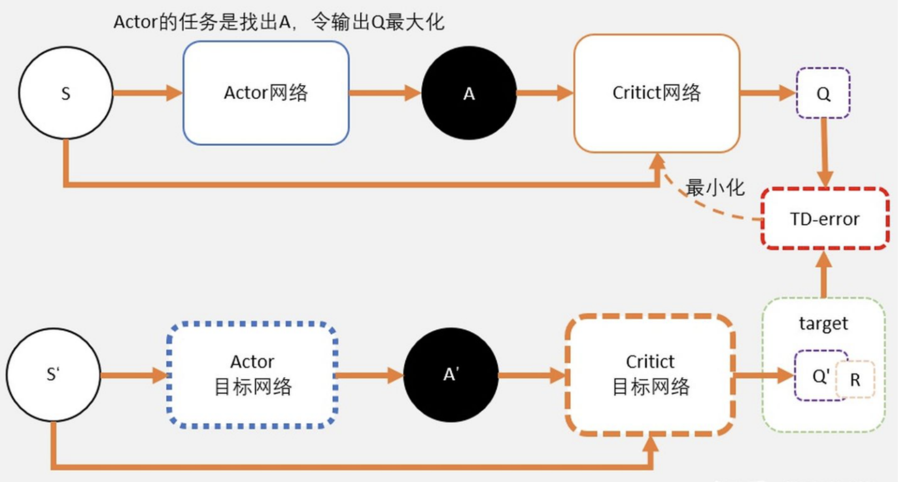
然后用梯度上升更新 θ ：

$$\theta \leftarrow \theta + \alpha' * pg$$

DDPG Framework



DDPG Framework (2)



DDPG Algorithm

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

DDPG Algorithm (2)

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**

行动策略为随机策略

   Select action $\boxed{a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t}$ according to the current policy and exploration noise
   Execute action a_t and observe reward r_t and observe new state s_{t+1}
   Store transition (s_t, a_t, r_t, s_{t+1}) in R
   Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R

经验回放

   Set $\boxed{y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})}$

目标网络

   Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q)^2)$
   Update the actor policy using the sampled gradient:

$$\nabla_{\theta^\mu}\mu|_{s_i} \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

   Update the target networks:

$$\boxed{\begin{aligned}\theta^{Q'} &\leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'} \\ \theta^{\mu'} &\leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}\end{aligned}}$$

目标网络参数更新

  end for
end for

Example: TORCS

https://youtu.be/8CNck-hdys8

Table of Contents

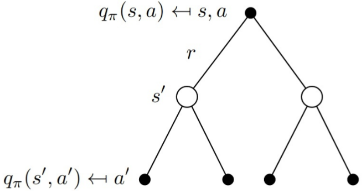
Soft Actor-Critic (SAC)

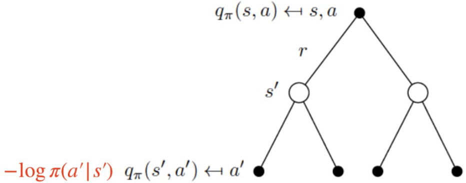
1. SAC optimizes a stochastic policy in an off-policy way, which unifies stochastic policy optimization and DDPG-style approaches
2. SAC incorporates **entropy regularization**
3. Entropy is a quantity which measures how random a random variable is, $H(P) = E_{x \sim P}[-\log P(x)]$
4. Entropy-regularized RL: the policy is trained to maximize a trade-off between expected return and entropy, a measure of randomness in the policy

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{(s_t, a_t) \sim \rho_\pi} [\sum_t R(s_t, a_t)]$$

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{(s_t, a_t) \sim \rho_\pi} [\sum_t \underbrace{R(s_t, a_t)}_{reward} + \alpha \underbrace{H(\pi(\cdot|s_t))}_{entropy}]$$

Soft Actor-Critic (SAC)





$$q_\pi(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s',a')$$

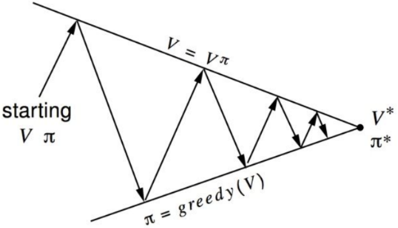
$$q_\pi(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \sum_{a' \in \mathcal{A}} \pi(a'|s')(q_\pi(s',a') - \alpha \log(\pi(a'|s'))$$

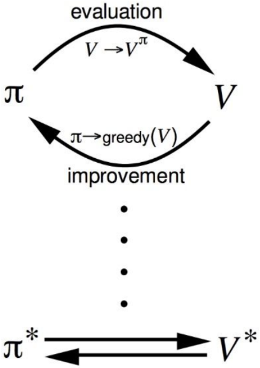
- The recursive Bellman equation for soft Q-function is

$$Q_{soft}(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}, a_{t+1}} [Q_{soft}(s_{t+1}, a_{t+1}) - \alpha \log(\pi(a_{t+1}|s_{t+1}))]$$

Soft Actor-Critic (SAC)

Policy Iteration





Policy evaluation Estimate v_π
 Iterative policy evaluation

Policy improvement Generate $\pi' \geq \pi$
 Greedy policy improvement

Soft Actor-Critic (SAC)

Policy iteration for traditional RL

- Policy evaluation: fix policy, update Q-function

$$Q_\pi(s,a) = r(s,a) + \lambda \mathbb{E}_{s',a'} Q_\pi(s',a')$$

- Policy improvement: update policy

$$\pi'(s) = \arg\max_a Q_\pi(s,a)$$

Policy iteration for SAC

- Policy evaluation: fix policy, update Q-function

$$Q^\pi_{soft}(s_t,a_t) = r(s_t,a_t) + \lambda \mathbb{E}_{s_{t+1},a_{t+1}}[Q^\pi_{soft}(s_{t+1},a_{t+1}) - \alpha \log(\pi(a_{t+1}|s_{t+1}))]$$

- Policy improvement: update policy

$$\pi' = \arg\min_{\pi_k \in \Pi} D_{KL}(\pi_k(\cdot|s_t)||\frac{\exp(\frac{1}{\alpha}Q^\pi_{soft}(s_t,\cdot))}{Z^\pi_{soft}(s_t)})$$