

Actor-Critic Policy Gradient

Spring, 2021

Outline

- 1 Policy Gradient Review
- 2 Actor-Critic Policy Gradient
- 3 Deterministic Policy Gradient

Table of Contents

- 1 Policy Gradient Review
- 2 Actor-Critic Policy Gradient
- 3 Deterministic Policy Gradient

Policy Gradient

Different Schools of Reinforcement Learning

- ① Value-based RL: solve RL through dynamic programming
 - ① Classic RL and control theory
 - ② Representative algorithms: Deep Q-learning and its variant
 - ③ Representative researchers: Richard Sutton (no more than 20 pages on PG out of the 500-page textbook), David Silver, from DeepMind
- ② Policy-based RL: solve RL mainly through learning
 - ① Machine learning and deep learning
 - ② Representative algorithms: PG, and its variants TRPO, PPO, and others
 - ③ Representative researchers: Pieter Abbeel, Sergey Levine, John Schulman, from OpenAI, Berkeley

Policy Gradient

- ① State-of-the-art RL methods are almost all policy-based
 - ① **A2C and A3C:** Asynchronous Methods for Deep Reinforcement Learning, ICML'16. Representative high-performance actor-critic algorithm: <https://openai.com/blog/baselines-acktr-a2c/>
 - ② **TRPO:** Schulman, L., Moritz, Jordan, Abbeel (2015). Trust region policy optimization: deep RL with natural policy gradient and adaptive step size
 - ③ **PPO:** Schulman, Wolski, Dhariwal, Radford, Klimov (2017). Proximal policy optimization algorithms: deep RL with importance sampled policy gradient

Policy Gradient

Review – Policy Gradient

$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} \left(\underbrace{\sum_{t'=t}^{T_n} \gamma^{t'-t} r_{t'}^n - b}_{G_t^n : \text{obtained via interaction}} \right) \nabla \log p_\theta(a_t^n | s_t^n)$$

baseline

Policy Gradient

Review – Policy Gradient

$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} \left(\underbrace{\sum_{t'=t}^{T_n} \gamma^{t'-t} r_{t'}^n - b}_{G_t^n : \text{ obtained via interaction}} \right) \nabla \log p_\theta(a_t^n | s_t^n)$$

baseline

G_t^n : obtained via interaction
Very unstable

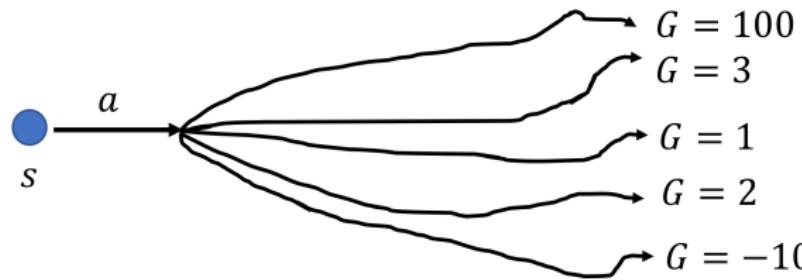
Policy Gradient

Review – Policy Gradient

$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} \left(\underbrace{\sum_{t'=t}^{T_n} \gamma^{t'-t} r_{t'}^n - b}_{G_t^n : \text{ obtained via interaction}} \right) \nabla \log p_\theta(a_t^n | s_t^n)$$

baseline

Very unstable



Policy Gradient

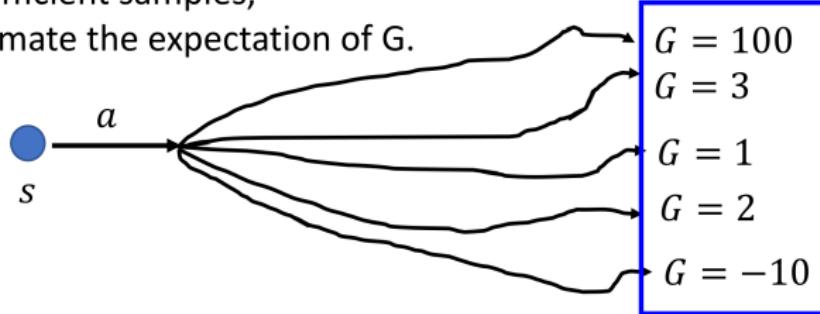
Review – Policy Gradient

$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} \left(\underbrace{\sum_{t'=t}^{T_n} \gamma^{t'-t} r_{t'}^n - b}_{G_t^n : \text{ obtained via interaction}} \right) \nabla \log p_\theta(a_t^n | s_t^n)$$

baseline

Very unstable

With sufficient samples,
approximate the expectation of G.



Policy Gradient

Review – Policy Gradient

$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} \left(\underbrace{\sum_{t'=t}^{T_n} \gamma^{t'-t} r_{t'}^n - b}_{G_t^n : \text{ obtained via interaction}} \right) \nabla \log p_\theta(a_t^n | s_t^n)$$

baseline

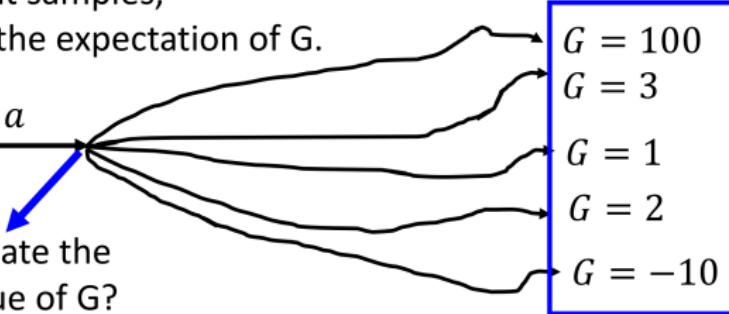
Very unstable

With sufficient samples,
approximate the expectation of G.



a
 s

Can we estimate the
expected value of G?



Policy Gradient

Review – Q-Learning

- State value function $V^\pi(s)$
 - When using actor π , the *cumulated* reward expects to be obtained after visiting state s
- State-action value function $Q^\pi(s, a)$
 - When using actor π , the *cumulated* reward expects to be obtained after taking a at state s

Policy Gradient

Review – Q-Learning

- State value function $V^\pi(s)$
 - When using actor π , the *cumulated* reward expects to be obtained after visiting state s
- State-action value function $Q^\pi(s, a)$
 - When using actor π , the *cumulated* reward expects to be obtained after taking a at state s

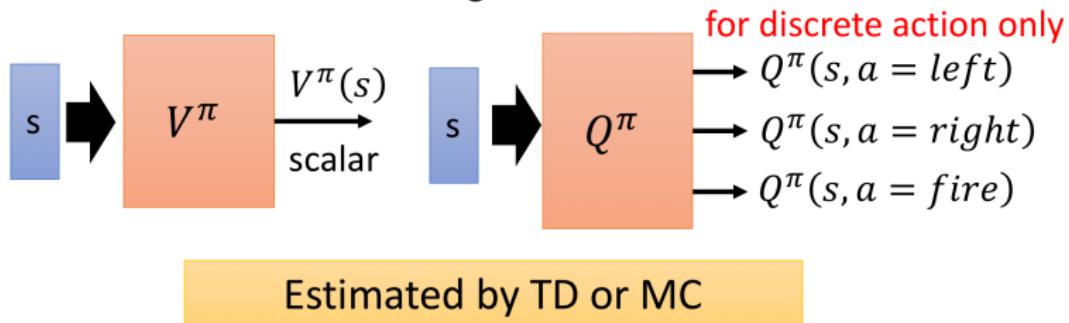


Table of Contents

- 1 Policy Gradient Review
- 2 Actor-Critic Policy Gradient
- 3 Deterministic Policy Gradient

Actor-Critic

Actor-Critic

$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} \left(\underbrace{\sum_{t'=t}^{T_n} \gamma^{t'-t} r_{t'}^n - b}_{G_t^n : \text{ obtained via interaction}} \right) \nabla \log p_\theta(a_t^n | s_t^n)$$

baseline

\downarrow

$$E[G_t^n] = Q^{\pi_\theta}(s_t^n, a_t^n)$$

Actor-Critic

Actor-Critic

$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} \left(\underbrace{\sum_{t'=t}^{T_n} \gamma^{t'-t} r_{t'}^n - b}_{G_t^n : \text{obtained via interaction}} \right) \nabla \log p_\theta(a_t^n | s_t^n)$$

↑
baseline
 $V^{\pi_\theta}(s_t^n)$

↓

$$E[G_t^n] = Q^{\pi_\theta}(s_t^n, a_t^n)$$

Actor-Critic

Actor-Critic

$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} \left(\sum_{t'=t}^{T_n} \gamma^{t'-t} r_{t'}^n - b \right) \nabla \log p_\theta(a_t^n | s_t^n)$$

G_t^n : obtained via interaction

$Q^{\pi_\theta}(s_t^n, a_t^n) - V^{\pi_\theta}(s_t^n)$

$V^{\pi_\theta}(s_t^n)$

baseline

$E[G_t^n] = Q^{\pi_\theta}(s_t^n, a_t^n)$

Actor-Critic

Advantage Actor-Critic (A2C)

$$Q^\pi(s_t^n, a_t^n) - V^\pi(s_t^n)$$

Estimate two networks? We can only estimate one.



Actor-Critic

Advantage Actor-Critic (A2C)

$$Q^\pi(s_t^n, a_t^n) - V^\pi(s_t^n)$$

Estimate two networks? We can only estimate one.

$$Q^\pi(s_t^n, a_t^n) = E[r_t^n + V^\pi(s_{t+1}^n)]$$

$$Q^\pi(s_t^n, a_t^n) = r_t^n + V^\pi(s_{t+1}^n)$$

Actor-Critic

Advantage Actor-Critic (A2C)

$$Q^\pi(s_t^n, a_t^n) - V^\pi(s_t^n)$$

Estimate two networks? We can only estimate one.



$$r_t^n + V^\pi(s_{t+1}^n) - V^\pi(s_t^n)$$

$$Q^\pi(s_t^n, a_t^n) = E[r_t^n + V^\pi(s_{t+1}^n)]$$

$$Q^\pi(s_t^n, a_t^n) = r_t^n + V^\pi(s_{t+1}^n)$$

Actor-Critic

Advantage Actor-Critic (A2C)

$$Q^\pi(s_t^n, a_t^n) - V^\pi(s_t^n)$$

Estimate two networks? We can only estimate one.



$$r_t^n + V^\pi(s_{t+1}^n) - V^\pi(s_t^n)$$

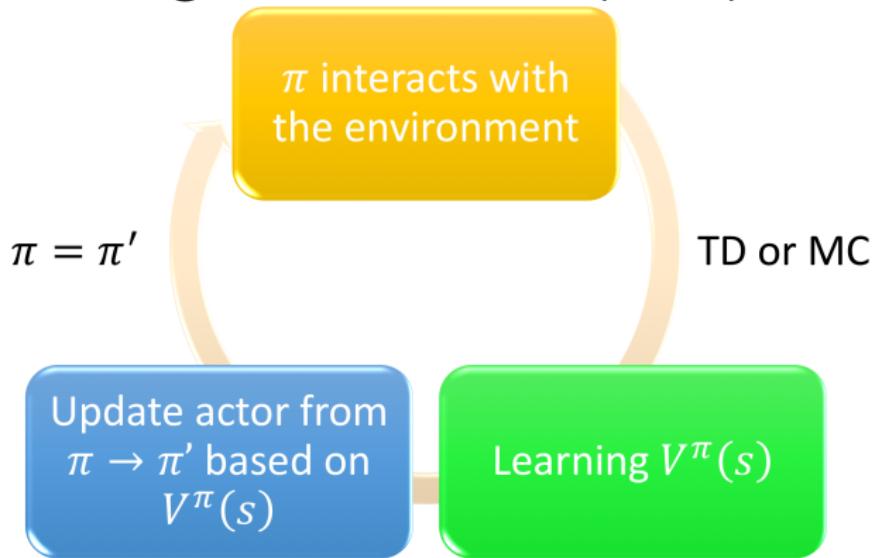
Only estimate state value
A little bit variance

$$Q^\pi(s_t^n, a_t^n) = E[r_t^n + V^\pi(s_{t+1}^n)]$$

$$Q^\pi(s_t^n, a_t^n) = r_t^n + V^\pi(s_{t+1}^n)$$

Actor-Critic

Advantage Actor-Critic (A2C)

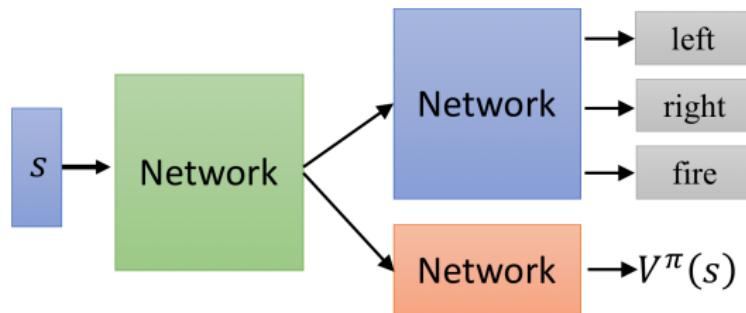


$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} (r_t^n + V^\pi(s_{t+1}^n) - V^\pi(s_t^n)) \nabla \log p_\theta(a_t^n | s_t^n)$$

Actor-Critic

Advantage Actor-Critic (A2C)

- Tips
 - The parameters of actor $\pi(s)$ and critic $V^\pi(s)$ can be shared



Actor-Critic

- Simple actor-critic algorithm based on action-value critic
- Using linear value function approximate $Q_w(s, a) = \phi(s, a)^T w$
 - Critic** Updates w by linear TD(0)
 - Actor** Updates θ by policy gradient

function QAC

 Initialise s, θ

 Sample $a \sim \pi_\theta$

for each step **do**

 Sample reward $r = \mathcal{R}_s^a$; sample transition $s' \sim \mathcal{P}_{s,a}^a$.

 Sample action $a' \sim \pi_\theta(s', a')$

$\delta = r + \gamma Q_w(s', a') - Q_w(s, a)$

$\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(s, a) Q_w(s, a)$

$w \leftarrow w + \beta \delta \phi(s, a)$

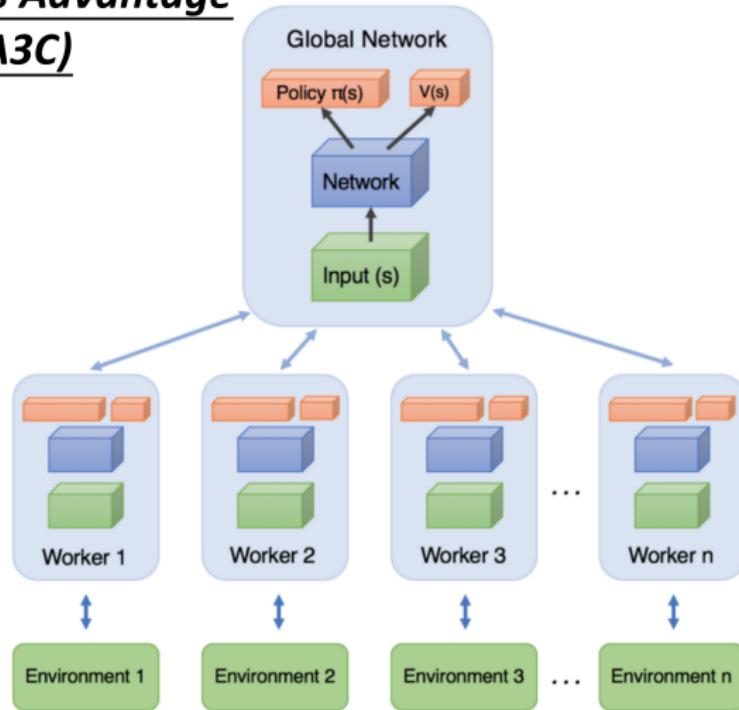
$a \leftarrow a', s \leftarrow s'$

end for

end function

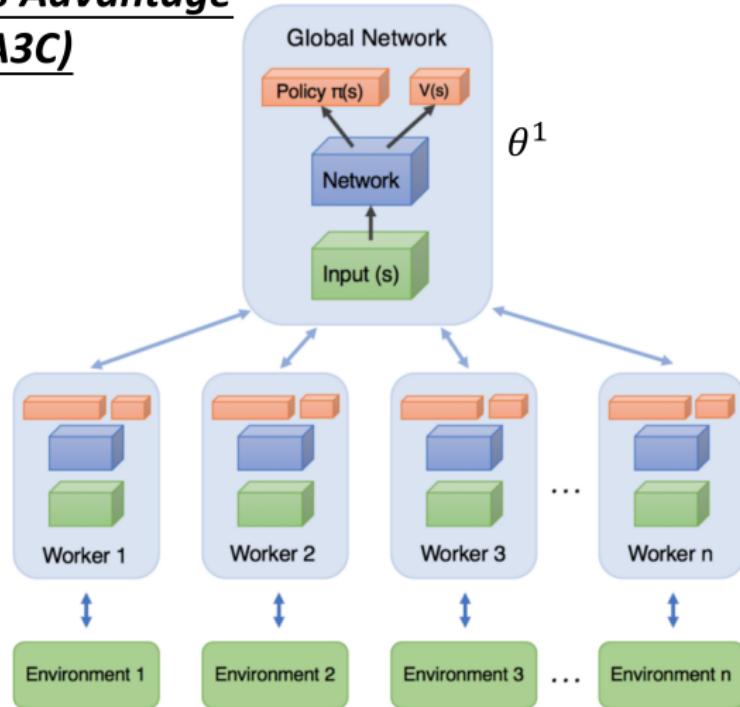
Actor-Critic

Asynchronous Advantage Actor-Critic (A3C)



Actor-Critic

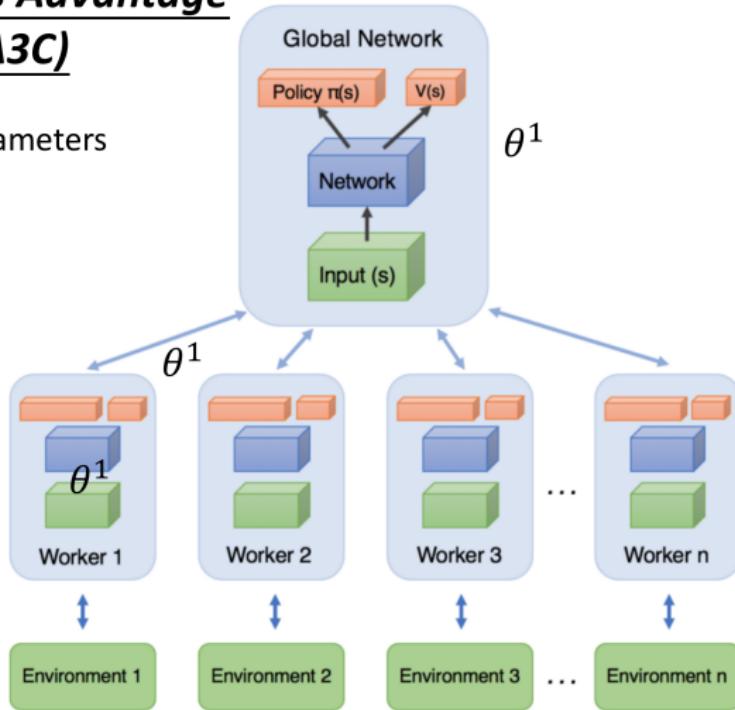
Asynchronous Advantage Actor-Critic (A3C)



Actor-Critic

Asynchronous Advantage Actor-Critic (A3C)

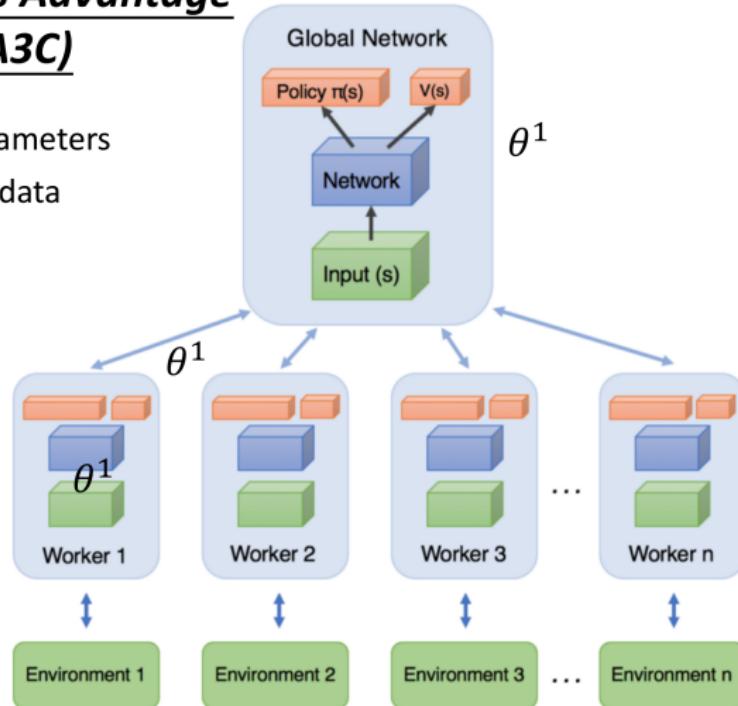
1. Copy global parameters



Actor-Critic

Asynchronous Advantage Actor-Critic (A3C)

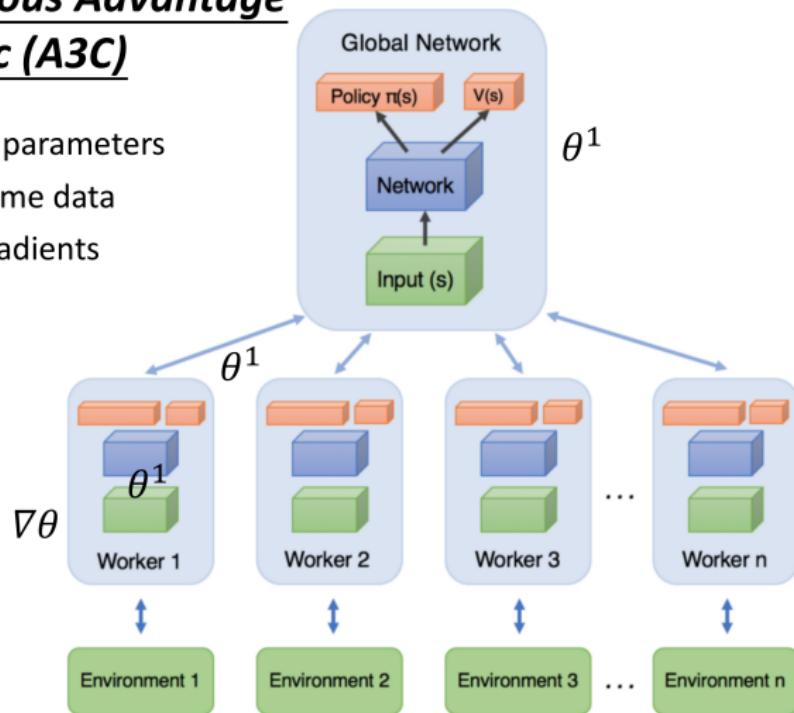
1. Copy global parameters
 2. Sampling some data



Actor-Critic

Asynchronous Advantage Actor-Critic (A3C)

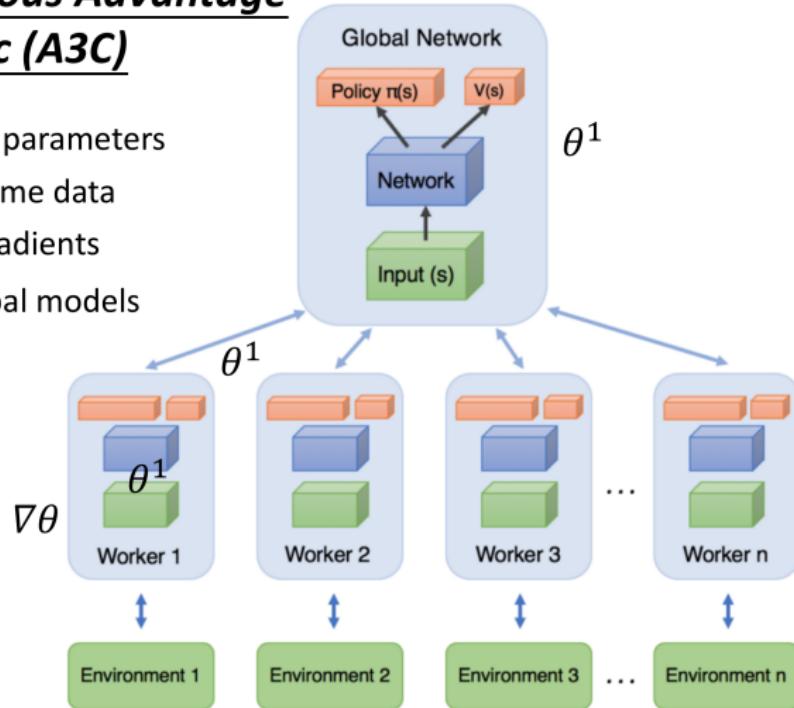
1. Copy global parameters
2. Sampling some data
3. Compute gradients



Actor-Critic

Asynchronous Advantage Actor-Critic (A3C)

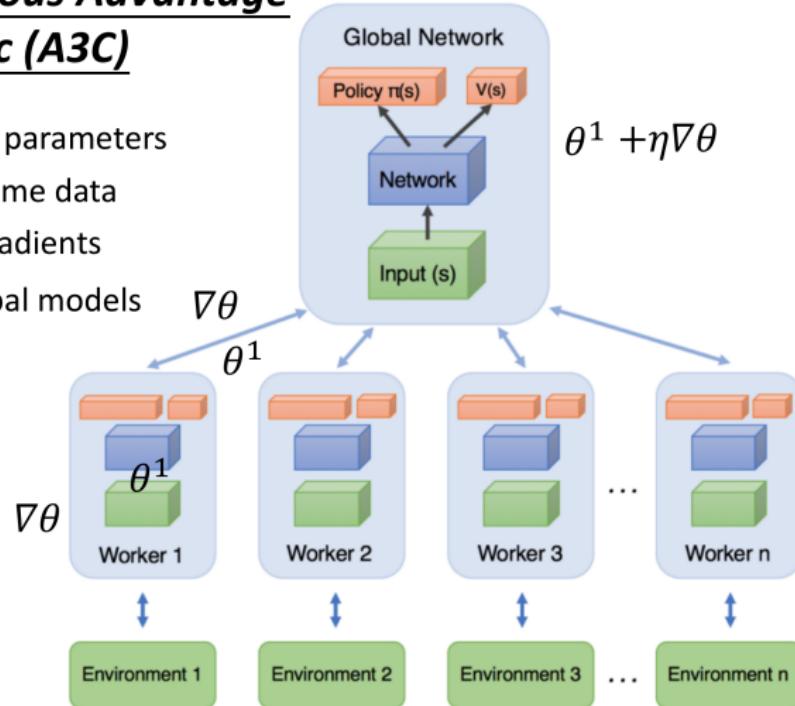
1. Copy global parameters
 2. Sampling some data
 3. Compute gradients
 4. Update global models



Actor-Critic

Asynchronous Advantage Actor-Critic (A3C)

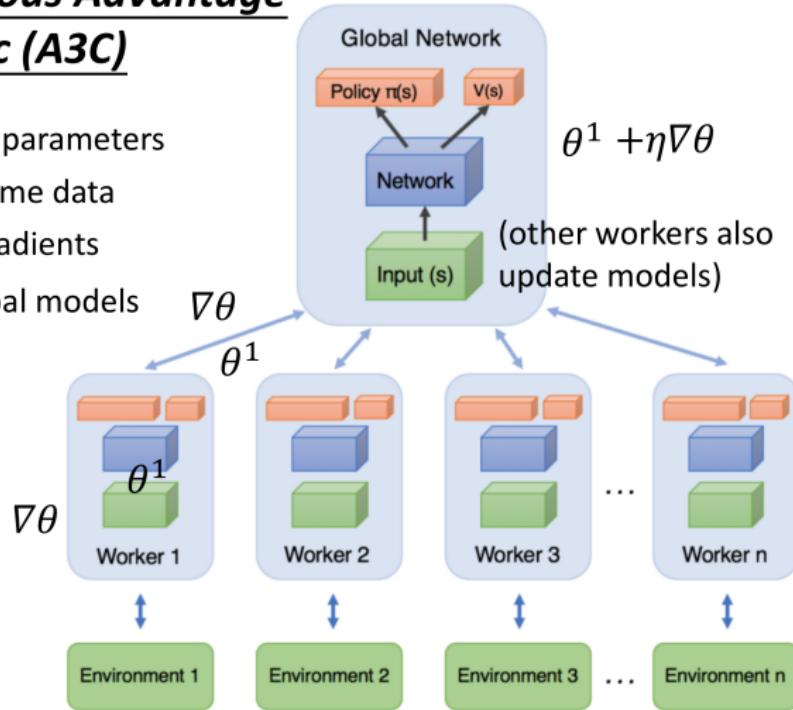
1. Copy global parameters
 2. Sampling some data
 3. Compute gradients
 4. Update global models



Actor-Critic

Asynchronous Advantage Actor-Critic (A3C)

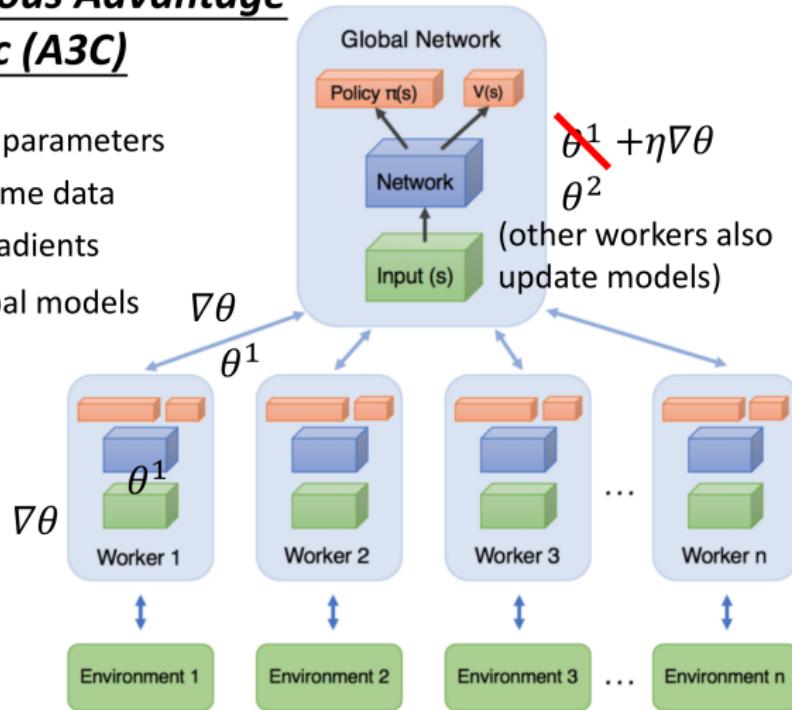
1. Copy global parameters
 2. Sampling some data
 3. Compute gradients
 4. Update global models



Actor-Critic

Asynchronous Advantage Actor-Critic (A3C)

1. Copy global parameters
 2. Sampling some data
 3. Compute gradients
 4. Update global models



Asynchronous Advantage Actor-Critic (A3C)

- Multiple asynchronous actors-learners run in parallel
- Explore different parts of environment, i.e., different exploration policies in different learners
- Stabilize learning, reduce training time, and without large resource requirements
- Add the entropy of policy to objective function
- Use standard non-centered RMSProp update

$$g = \alpha g + (1 - \alpha) \Delta \theta^2 \text{ and } \theta \leftarrow \theta - \eta \frac{\Delta \theta}{\sqrt{g + \epsilon}}$$

Asynchronous Methods for Deep Reinforcement Learning,
Proceedings of ICML, 2016

A3C Algorithm

Algorithm S3 Asynchronous advantage actor-critic - pseudocode for each actor-learner thread.

//Assume global shared parameter vectors θ and θ_v , and global shared counter $T = 0$

//Assume thread-specific parameter vectors θ' and θ'_v

Initialize thread step counter $t \leftarrow 1$

repeat

 Reset gradients: $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$.

 Synchronize thread-specific parameters $\theta' = \theta$ and $\theta'_v = \theta_v$

$t_{start} = t$

 Get state s_t

repeat

 Perform a_t according to policy $\pi(a_t | s_t; \theta')$

 Receive reward r_t and new state s_{t+1}

$t \leftarrow t + 1$

$T \leftarrow T + 1$

until terminal s_t **or** $t - t_{start} == t_{max}$

$R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t, \theta'_v) & \text{for non-terminal } s_t // \text{Bootstrap from last state} \end{cases}$

for $i \in \{t - 1, \dots, t_{start}\}$ **do**

$R \leftarrow r_i + \gamma R$

 Accumulate gradients wrt θ' : $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i | s_i; \theta') (R - V(s_i; \theta'_v))$

 Accumulate gradients wrt θ'_v : $d\theta_v \leftarrow d\theta_v + \partial (R - V(s_i; \theta'_v))^2 / \partial \theta'_v$

end for

 Perform asynchronous update of θ using $d\theta$ and of θ_v using $d\theta_v$.

until $T > T_{max}$

Summary of Policy Gradient Algorithms

- The **policy gradient** has many equivalent forms

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(s, a) \ v_t] \quad \text{REINFORCE}$$

$$= \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(s, a) \ Q^w(s, a)] \quad \text{Q Actor-Critic}$$

$$= \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(s, a) \ A^w(s, a)] \quad \text{Advantage Actor-Critic}$$

$$= \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(s, a) \ \delta] \quad \text{TD Actor-Critic}$$

$$= \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(s, a) \ \delta e] \quad \text{TD}(\lambda) \text{ Actor-Critic}$$

$$G_{\theta}^{-1} \nabla_{\theta} J(\theta) = w \quad \text{Natural Actor-Critic}$$

- Each leads a stochastic gradient ascent algorithm
- Critic uses **policy evaluation** (e.g. MC or TD learning) to estimate $Q^{\pi}(s, a)$, $A^{\pi}(s, a)$ or $V^{\pi}(s)$

Table of Contents

- 1 Policy Gradient Review
- 2 Actor-Critic Policy Gradient
- 3 Deterministic Policy Gradient

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

Introduction

- It was previously believed that the deterministic policy gradient did not exist.
- Or could only be obtained when using a model.
- Until 2014, deterministic policy gradient was proposed, and showed that it has a simple model-free form.
- In 2016, an actor-critic algorithm based on deterministic policy gradient was proposed for continuous action spaces.

Deterministic Policy Gradient Algorithms, ICML 2014

Continuous Control with Deep Reinforcement Learning, ICLR 2016

Stochastic Policy

- The policy is **stochastic** and denoted by

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

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.

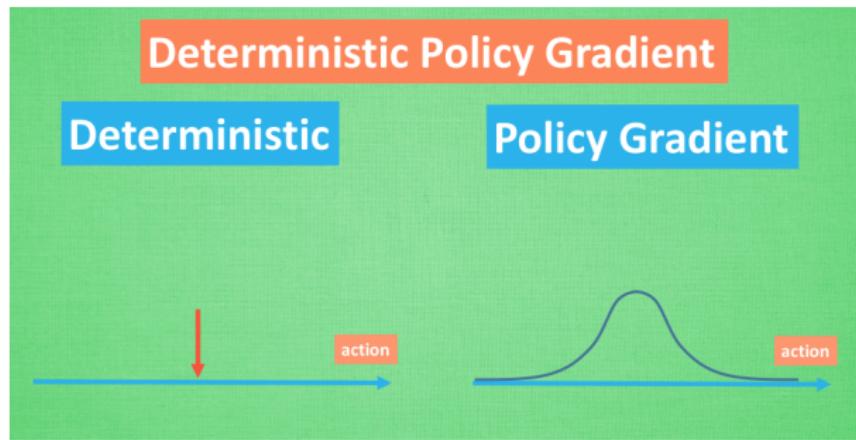
- $\pi_\theta(a_t|s_t)$ is the conditional **probability density** at a_t associated with the policy.
- For the same state s , the actions stochastically selected according to θ might be different.

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



Deterministic Policy vs. Stochastic Policy

- For *stochastic* case: the policy gradient integrates over both state and action spaces

$$\begin{aligned}\nabla_{\theta} J(\pi_{\theta}) &= \int_{\mathcal{S}} \rho^{\pi}(s) \int_{\mathcal{A}} \nabla_{\theta} \pi_{\theta}(a|s) Q^{\pi}(s, a) da ds \\ &= \mathbb{E}_{s \sim \rho^{\pi}, a \sim \pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a|s) Q^{\pi}(s, a)]\end{aligned}$$

- For *deterministic* case: the policy gradient only integrates over the state spaces since the action selections are deterministic

$$\begin{aligned}\nabla_{\theta} J(\mu_{\theta}) &= \int_{\mathcal{S}} \rho^{\mu}(s) \nabla_{\theta} \mu_{\theta}(s) \left. \nabla_a Q^{\mu}(s, a) \right|_{a=\mu_{\theta}(s)} ds \\ &= \mathbb{E}_{s \sim \rho^{\mu}} \left[\nabla_{\theta} \mu_{\theta}(s) \left. \nabla_a Q^{\mu}(s, a) \right|_{a=\mu_{\theta}(s)} \right]\end{aligned}$$

Deterministic Policy vs. Stochastic Policy (2)

- Hence, computing the stochastic policy gradient may require more samples.
- Especially if the action space has many dimensions.
- It is shown that the deterministic policy gradient is the **limiting case**, as policy variance tends to zero, of the stochastic policy gradient.

Generalized Policy Iteration

The majority of model-free reinforcement learning algorithms are based on generalized policy iteration: interleaving policy evaluation with policy improvement.

- Policy evaluation methods estimate the action-value function

$$Q^\mu(s, a)$$

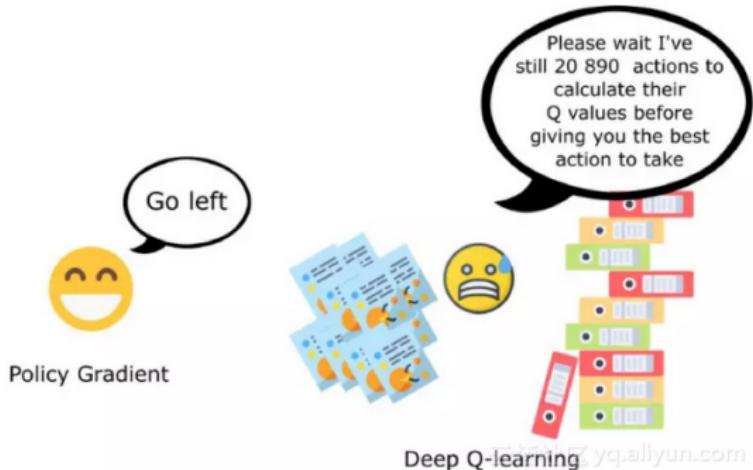
- e.g. Monte-Carlo evaluation or TD learning
- Policy improvement methods update the policy w.r.t the $Q^\mu(s, a)$
 - the most common approach: greedy maximization

$$\mu^{k+1}(s) = \arg \max_a Q^{\mu^k}(s, a)$$

- or soft maximization, etc.

Deterministic Policy Improvement

- In continuous action spaces, greedy policy improvement becomes problematic, requiring a global maximization at every step



- Instead, moving the policy in the direction of the gradient of Q , rather than globally maximizing Q

Deterministic Policy Improvement (2)

- For each visited state s , the policy parameter θ^{k+1} are updated **in proportion to** the gradient $\nabla_{\theta} Q^{\mu^k}(s, \mu_{\theta}(s))$
- Each state suggests a **different** direction of policy improvement
- By taking an expectation w.r.t. the state distribution ρ^{μ^k}

$$\theta^{k+1} = \theta^k + \alpha \mathbb{E}_{s \sim \rho^{\mu^k}} [\nabla_{\theta} Q^{\mu^k}(s, \mu_{\theta}(s))]$$

- Policy improvement is decomposed into the gradient of action-value w.r.t. actions, and the gradient of policy w.r.t. policy parameters

$$\theta^{k+1} = \theta^k + \alpha \mathbb{E}_{s \sim \rho^{\mu^k}} \left[\nabla_{\theta} \mu_{\theta}(s) \nabla_a Q^{\mu^k}(s, a) \Big|_{a=\mu_{\theta}(s)} \right]$$

- By convention $\nabla_{\theta} \mu_{\theta}(s)$ is a *Jacobian matrix* such that each column is the gradient $\nabla_{\theta} [\mu_{\theta}(s)]_d$ of the d -th action dimension of the policy w.r.t. θ

Guarantee of Policy Improvement

- If change the policy, different states are visited and the state distribution ρ^{μ^k} will **change**

$$\theta^{k+1} = \theta^k + \alpha \mathbb{E}_{s \sim \rho^{\mu^k}} [\nabla_{\theta} \mu_{\theta}(s) \nabla_a Q^{\mu^k}(s, a) \Big|_{a=\mu_{\theta}(s)}]$$

- This policy improvement method **cannot guarantee improvement** without taking account of the change to distribution
- recall the *(Stochastic) Policy Gradient Theorem*, there is no need to compute the gradient of the state distribution.

(Stochastic) Policy Gradient Theorem

For any differentiable policy $\pi_{\theta}(s, a)$, for any of the policy objective functions, the policy gradient is

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(s, a) Q^{\pi_{\theta}}(s, a)]$$

Deterministic Policy Objective Function

Analogously to the stochastic case →

- Consider a deterministic policy $\mu_\theta : \mathcal{S} \rightarrow \mathcal{A}$ w.r.t. $\theta \in \mathbb{R}^n$
- Define a **deterministic** policy objective function

$$J(\mu_\theta) = \mathbb{E}[r_1^\gamma | \mu]$$

- Probability distribution $p(s \rightarrow s', t, \mu)$ and discounted state distribution $\rho^\mu(s)$

$$\begin{aligned} J(\mu_\theta) &= \int_{\mathcal{S}} \rho^\mu(s) r(s, \mu_\theta(s)) ds \\ &= \mathbb{E}_{s \sim \rho^\mu} [r(s, \mu_\theta(s))] \end{aligned}$$

Deterministic Policy Gradient Theorem

- Replace the immediate reward $r(s, \mu_\theta(s))$ with $Q^\mu(s, \mu_\theta(s))$

Deterministic Policy Gradient Theorem

Suppose that the MDP satisfies that

$p(s'|s, a)$, $\nabla_a p(s'|s, a)$, $\mu_\theta(s)$, $\nabla_\theta \mu_\theta(s)$, $r(s, a)$, $\nabla_a r(s, a)$, $p_1(s)$ are continuous in all parameters and variables s, a, s' . Then,

$$\begin{aligned}\nabla_\theta J(\mu_\theta) &= \int_{\mathcal{S}} \rho^\mu(s) \nabla_\theta \mu_\theta(s) \nabla_a Q^{\mu^k}(s, a)|_{a=\mu_\theta(s)} ds \\ &= \mathbb{E}_{s \sim \rho^\mu} [\nabla_\theta \mu_\theta(s) \nabla_a Q^{\mu^k}(s, a)|_{a=\mu_\theta(s)}]\end{aligned}$$

Off-Policy Deterministic Actor-Critic

- Behaving according to a deterministic policy will not ensure adequate exploration, and may lead to sub-optimal solutions
- For adequate exploration, off-policy learning is employed
- A actor learns a deterministic target policy $\mu_\theta(s)$ from trajectories generated by an *arbitrary stochastic* behaviour policy $\beta(a|s)$
- A critic estimates the action-value function $Q^w(s, a) \approx Q^\mu(s, a)$, off-policy from trajectories generated by $\beta(a|s)$
 - use Q-learning to estimate the action-value function

Off-Policy Deterministic Actor

- First, we modify the policy objective function to be the value function of the *target policy* $\mu_\theta(s)$, **averaged over** the *state distribution* of the *behaviour policy* $\beta_\theta(a|s)$,

$$\begin{aligned} J_\beta(\mu_\theta) &= \int_{\mathcal{S}} \rho^\beta(s) V^\mu(s) ds \\ &= \int_{\mathcal{S}} \rho^\beta(s) Q^\mu(s, \mu_\theta(s)) ds \end{aligned}$$

$$\begin{aligned} \nabla_\theta J_\beta(\mu_\theta) &\approx \int_{\mathcal{S}} \rho^\beta(s) \nabla_\theta \mu_\theta(s) Q^\mu(s, a) ds \\ &= \mathbb{E}_{s \sim \rho^\beta} [\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s, a) \Big|_{a=\mu_\theta(s)}] \end{aligned}$$

Deterministic Actor-Critic Algorithm

$$\begin{aligned}\delta_t &= r_t + \gamma Q^w(s_{t+1}, \mu_\theta(s_{t+1})) - Q^w(s_t, a_t) \\ w_{t+1} &= w_t + \alpha_w \delta_t \nabla_w Q^w(s_t, a_t) \\ \theta_{t+1} &= \theta_t + \alpha_\theta \nabla_\theta \mu_\theta(s_t) \nabla_a Q^w(s_t, a_t) |_{a=\mu_\theta(s)}\end{aligned}$$

Compatible Function Approximation

- Find a compatible function approximator $Q^w(s, a)$ such that the true gradient is preserved.

Theorem

A function approximator $Q^w(s, a)$ is compatible with a deterministic policy $\mu_\theta(s)$, $\nabla_\theta J_\beta(\theta) = \mathbb{E}[\nabla_\theta \mu_\theta(s) \nabla_a Q^w(s, a)|_{a=\mu_\theta(s)}]$, if

- $\nabla_a Q^w(s, a)|_{a=\mu_\theta(s)} = \nabla_\theta \mu_\theta(s)^T w$ and

- w minimizes the mean-squared error,

$$MSE(\theta, w) = \mathbb{E}[\epsilon(s; \theta, w)^T \epsilon(s; \theta, w)] \text{ where}$$

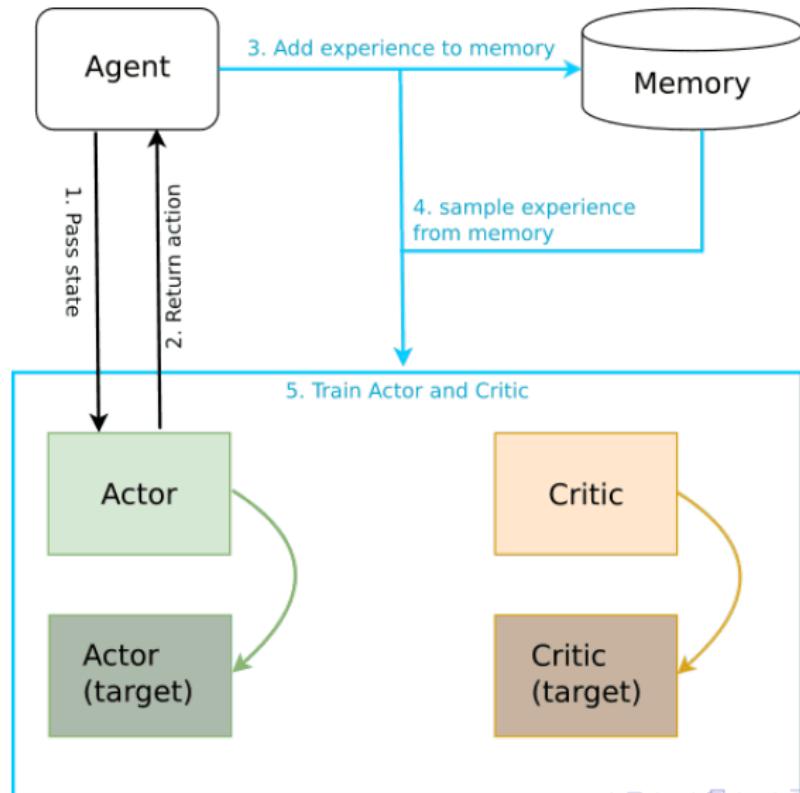
$$\epsilon(s; \theta, w) = \nabla_a Q^w(s, a)|_{a=\mu_\theta(s)} - \nabla_a Q^\mu(s, a)|_{a=\mu_\theta(s)}$$

Deep DPG (DDPG)

- DDPG is a kind of off-policy actor-critic algorithm using deep function approximators to learn policies in high-dimensional, continuous action spaces.
- The experience replay idea of DQN is introduced to learn across a set of uncorrelated transitions.
- Two sets of networks: critic network $Q(s, a|\theta^Q)$ and actor network $\mu(s|\theta^\mu)$, with weights θ^Q and θ^μ target network Q' and μ' , with weights $\theta^{Q'}$ and $\theta^{\mu'}$ updated by:

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

DDPG Framework



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 $a_t = \mu(s_t | \theta^\mu) + \mathcal{N}_t$ 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 $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1} | \theta^{\mu'}))$ 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:

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

Update the target networks:

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

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

目标网络参数更新

end for

end for

Example: TORCS

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