

Lecture 20 - Interconnections between policy and value

Guiliang Liu

The Chinese University of Hong Kong, Shenzhen

DDA4230: Reinforcement Learning
Course Page: [\[Click\]](#)

Recap: REINFORCE Algorithm

To compute the gradient $\nabla_{\theta} \mathbf{J}(\theta)$ algorithmically, we can **sample N trajectories** following the policy π and **use the empirical mean** to estimate the gradient

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

- For $Q^{\pi}(s, a)$, we can use return $G_t = \sum \gamma^t r_t$ to estimate.
- For $\nabla_{\theta} \log \pi_{\theta}(a | s)$, it depends on the form of the policy.



Recap: REINFORCE Algorithm

Algorithm 1: REINFORCE (Monte-Carlo method)

Initialize the policy parameter θ

for each episode do

 Sample one trajectory on policy π_θ : $s_0, a_0, r_0, s_1, a_1, \dots, s_T$

for each $t = 0, 1, \dots, T$ **do**

$G_t \leftarrow \sum_{t'=t}^T \gamma^{t'-t} r_{t'}$

$\theta \leftarrow \theta + \alpha \gamma^t G_t \nabla_\theta \log \pi_\theta(a_t | s_t)$



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Actor Critic methods

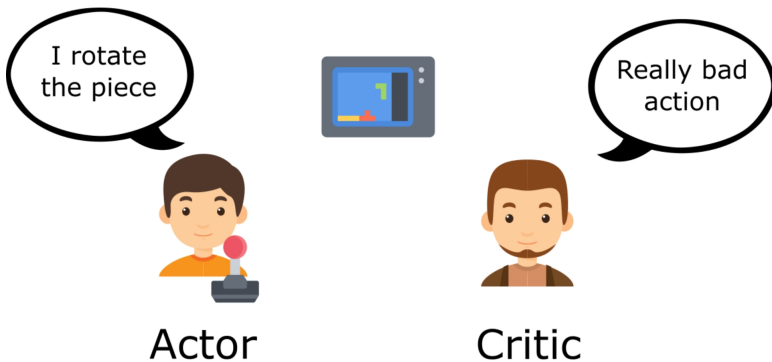
Motivation of Actor-critic.

- Most of the variance is from the Monte-Carlo estimation G_t of $Q(s_t, a_t)$.
- We can estimate parametrized $Q(s, a)$ and bootstrap the estimation into the policy gradient. This results in a biased estimator but with a much lower variance.
- One way to estimate the value function is the temporal-difference method, With this bootstrap, the method is called actor-critic.



Actor Critic methods

An illustrative example of the idea of actor-critic.



Actor Critic methods

Actor-critic methods consist of two models.

- The **critic updates the value function** parameters w .
- The **actor updates the policy parameters** θ in the direction suggested by the critic.

Note that although the REINFORCE with baseline method learns both a policy and a state value function, we do not consider it to be an actor-critic method because its state value function is used only as a baseline instead of a critic.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Actor Critic methods

One-step actor-critic methods 1) replace the full return of REINFORCE with the one-step return and 2) use a learned state value function as the baseline, as

$$\begin{aligned}\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \alpha_{\theta}(G_t - \hat{V}(s_t, \mathbf{w}))\nabla \log \pi_{\theta}(a_t | s_t) \\ &= \boldsymbol{\theta}_t + \alpha_{\theta}(r_t + \gamma \hat{V}(s_{t+1}, \mathbf{w}) - \hat{V}(s_t, \mathbf{w}))\nabla \log \pi_{\theta}(a_t | s_t).\end{aligned}$$

This algorithm then takes two inputs: a differentiable policy parametrized by $\pi_{\theta}(a | s)$ and a differentiable state value function parametrized by $\hat{V}(s, \mathbf{w})$.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Actor Critic methods

Algorithm 3: One-step actor-critic (episodic)

Initialize the policy parameter θ and w at random. **for each episode do**

 Initialize s_0 , the first state of each episode

for each $t = 0, 1, \dots, T - 1$ **do**

 sample $a \sim \pi(a | s_t, \theta)$

 take action a and observe s', r

$\delta \leftarrow r + \gamma \hat{V}(s', w) - \hat{V}(s, w)$

$w \leftarrow w + \alpha_w \delta \nabla_w \hat{V}(s, w)$

$\theta \leftarrow \theta + \alpha_\theta \delta \nabla_\theta \log \pi(a | s, \theta)$

$s' \leftarrow s$



Actor Critic methods

Some advancement on the AC algorithm.

- **Advantages Actor Critic (A2C)** replace $r_t + \gamma \hat{V}(s_{t+1}, \mathbf{w})$ with $Q(s, a, \mathbf{w}')$, so that $A(s, a) = Q(s, a, \mathbf{w}') - \hat{V}(s_t, \mathbf{w})$.
- **Asynchronous Advantages Actor Critic (A3C)** asynchronously executes multiple agents in parallel, thereby de-correlating the agents's data and stabilizing training.

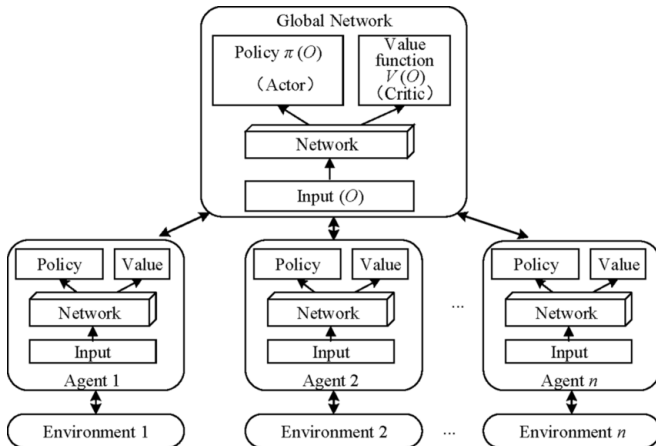


香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Actor Critic methods

The structure of Asynchronous Advantages Actor-Critic (A3C)



Soft Actor-Critic

Motivation. The combination of **off-policy learning** and **high-dimensional, nonlinear function approximation** with neural networks presents a major challenge for **stability and convergence**.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Soft Actor-Critic

Soft Actor-Critic is an **off-policy maximum entropy actor-critic algorithm**.

- This algorithm extends readily to very complex, high-dimensional tasks, where off-policy methods typically struggle to obtain good results.
- SAC also avoids the complexity and potential instability associated with approximate inference in prior off-policy maximum entropy algorithms.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Soft Policy Iteration

Soft Policy Iteration considers a **maximum entropy objective based on Standard RL**, which favors stochastic policies by augmenting the maximizing the expected sum of rewards objective with the expected entropy of the policy over $\rho_{\pi}(s_t)$, as

$$J(\pi) = \sum_{t=0}^T \mathbb{E}_{s_t, a_t, t \geq 0} [r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot | s_t))]. \quad (1)$$

The temperature parameter α determines the relative importance of the entropy term against the reward, and thus controls the stochasticity of the optimal policy (**encourage exploration**).¹



¹For the rest of this lecture notes, we will omit writing α .

Soft Policy Iteration

We will begin by deriving soft policy iteration based on our objective. For a fixed policy, the soft Q-value can be computed iteratively, starting from any function $Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ and repeatedly applying a modified Bellman backup operator \mathcal{B}^π given by

$$\mathcal{B}^\pi Q(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \mathbb{P}}[V(s_{t+1})], \quad (1)$$

$$V(s_t) = \mathbb{E}_{a_t \sim \pi}[Q(s_t, a_t) - \log \pi(a_t | s_t)] \quad (2)$$

is the soft state value function. We can obtain the soft value function for any policy π by repeatedly applying \mathcal{B}^π as formalized below.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Soft Policy Iteration

Lemma (Soft policy evaluation)

Consider the soft Bellman backup operator \mathcal{B}^π and a mapping $Q^0 : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ with $|\mathcal{A}| < \infty$, and define $Q^{k+1} = \mathcal{B}^\pi Q^k$. Then the sequence Q^k will converge to the soft Q -value of π as $k \rightarrow \infty$.

Proof.

Define the entropy augmented reward as $r_\pi(s_t, a_t) = r(s_t, a_t) + \mathbb{E}_{s_{t+1} \sim \mathbb{P}} [\mathcal{H}(\pi(\cdot | \mathbb{P}))]$ and rewrite the update rule as: $Q(s_t, a_t) \leftarrow r_\pi(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \mathbb{P}, a_{t+1} \sim \pi} [Q(s_{t+1}, a_{t+1})]$ and apply the standard convergence results for policy evaluation. □



Soft Policy Iteration

In the policy improvement step, the policy update results in an improved policy in terms of its soft value. For each state, we update the policy according to

$$\pi_{\text{new}} = \arg \min_{\pi' \in \Pi} d_{\text{KL}}(\pi'(\cdot | s_t) \| \frac{\exp(Q^{\pi_{\text{old}}}(s_t, \cdot))}{Z^{\pi_{\text{old}}}(s_t)}). \quad (1)$$

The partition function $Z^{\pi_{\text{old}}}(s_t)$ normalizes the distribution.

Lemma (Soft policy improvement)

Let $\pi_{\text{old}} \in \Pi$ and let π_{new} be the optimum of the minimization problem defined . Then, $Q^{\pi_{\text{new}}}(s_t, a_t) \geq Q^{\pi_{\text{old}}}(s_t, a_t)$ for all $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$ when $|\mathcal{A}| < \infty$.



Soft Policy Iteration

The full soft policy iteration algorithm alternates between the **soft policy evaluation** and the **soft policy improvement steps**, and it will provably converge to the **optimal maximum entropy policy** among the policies in Π , as shown in the below lemma.

Lemma (Soft policy iteration)

Repeated application of soft policy evaluation and soft policy improvement from any $\pi \in \Pi$ converges to a policy π^ such that $Q^{\pi^*}(s_t, a_t) \geq Q^\pi(s_t, a_t)$ for all $\pi \in \Pi$ and $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$, assuming that $|\mathcal{A}| < \infty$.*



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Soft Policy Iteration

Although this algorithm will provably find the optimal solution, we can perform it in its exact form only in the tabular case. Therefore, we will next approximate the algorithm for **continuous domains**, where we need to **rely on a function approximator to represent the Q-values**, and running the two steps until convergence would be computationally too expensive.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

SAC Algorithm for Deep RL

We will consider a parameterized **state value function** $V_\psi(s_t)$, **soft Q-function** $Q_\theta(s_t, a_t)$, and a **tractable policy** $\pi_\phi(a_t | s_t)$.

Update $V_\psi(s)$. The soft value function is trained to minimize the squared residual error

$$J_V(\psi) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[\frac{1}{2} (V_\psi(s_t) - \mathbb{E}_{a_t \sim \pi_\phi} [Q_\theta(s_t, a_t) - \log \pi_\phi(a_t | s_t)])^2 \right], \quad (1)$$

where \mathcal{D} is a replay buffer. The gradient can be estimated with an unbiased estimator

$$\hat{\nabla}_\psi J_V(\psi) = \nabla_\psi V_\psi(s_t) (V_\psi(s_t) - Q_\theta(s_t, a_t) + \log \pi_\phi(a_t | s_t)), \quad (2)$$

where the actions are sampled according to the current policy instead of the replay buffer and $Q_\theta(s_t, a_t)$ can be replaced by Monte-Carlo sample.



SAC Algorithm for Deep RL

Update $Q_\theta(s, a)$. The soft Q-function parameters can be trained to minimize:

$$J_Q(\theta) = \mathbb{E}_{(s_t, a_t) \sim \mathcal{D}} \left[\frac{1}{2} \left(Q_\theta(s_t, a_t) - \hat{Q}(s_t, a_t) \right)^2 \right], \hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \mathbb{P}} [V_{\bar{\psi}}(s_{t+1})],$$

which again can be optimized with stochastic gradients

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(s_t, a_t) \left(Q_\theta(s_t, a_t) - r(s_t, a_t) - \gamma V_{\bar{\psi}}(s_{t+1}) \right).$$

The update makes use of a target value network $V_{\bar{\psi}}$, where $\bar{\psi}$ can be 1) an exponentially moving average of the value network weights or 2) updated to match the current value function weights periodically.



SAC Algorithm for Deep RL

Update $\pi_\phi(a | s)$. Finally, the policy parameters can be learned by directly minimizing the expected KL-divergence:

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[d_{KL}(\pi_\phi(\cdot | s_t) \parallel \frac{\exp(Q_\theta(s_t, \cdot))}{Z_\theta(s_t)}) \right].$$

for minimizing J_π , we reparameterize the policy using a neural network transformation

$$a_t = f_\phi(\varepsilon_t; s_t),$$

where ε_t is an input noise vector, sampled from some fixed distribution, such as a spherical **Gaussian**.



SAC Algorithm for Deep RL

We can now rewrite the objective as

$$J_{\pi}(\phi) = \mathbb{E}_{s_t \sim \mathcal{D}, \varepsilon_t \sim \mathcal{N}} [\log \pi_{\phi}(f_{\phi}(\varepsilon_t; s_t) | s_t) - Q_{\theta}(s_t, f_{\phi}(\varepsilon_t; s_t))] ,$$

where π_{ϕ} is defined implicitly in terms of f_{ϕ} . We can approximate the gradient with

$$\widehat{\nabla_{\phi} J_{\pi}(\phi)} = \nabla_{\phi} \log \pi_{\phi}(a_t | s_t) + (\nabla_{a_t} \log \pi_{\phi}(a_t | s_t) - \nabla_{a_t} Q(s_t, a_t)) \nabla_{\phi} f_{\phi}(\varepsilon_t; s_t),$$

where a_t is evaluated at $f_{\phi}(\varepsilon_t; s_t)$. This unbiased gradient estimator extends the DDPG style policy gradients to any tractable stochastic policy.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

SAC Algorithm for Deep RL

Algorithm 4: Soft actor-critic (SAC)

Initialize parameter vectors $\psi, \bar{\psi}, \theta_1, \theta_2, \phi$

for each iteration do

for each environment step do

$$a_t \sim \pi_\phi(a_t | s_t)$$

$$s_{t+1} \sim \mathbb{P}(s_{t+1} | s_t, a_t)$$

$$\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1})\}$$

for each gradient step do

$$\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$$

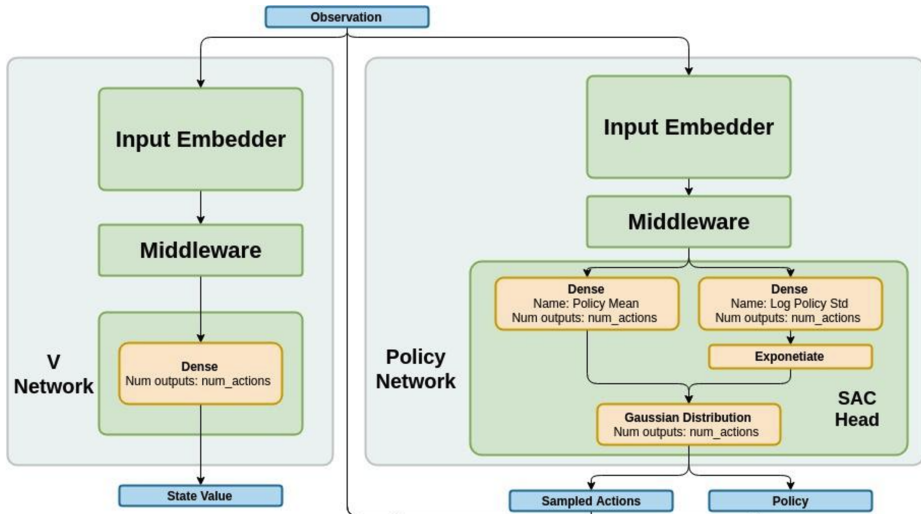
$$\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i) \text{ for } i \in \{1, 2\}$$

$$\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$$

$$\bar{\psi} \leftarrow \tau\psi + (1 - \tau)\bar{\psi}$$

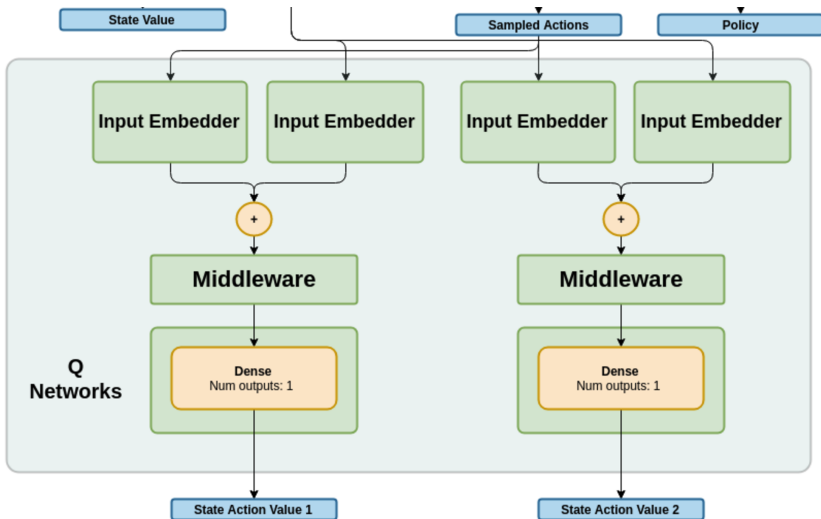


SAC Algorithm for Deep RL

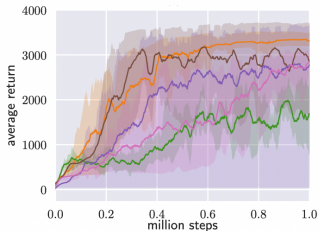


圳)
g Kong, Shenzhen

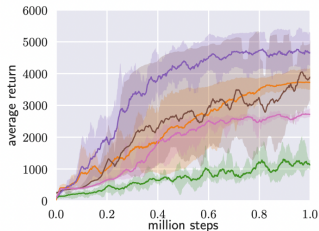
SAC Algorithm for Deep RL



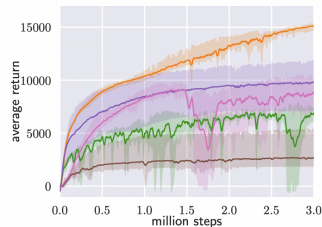
SAC Algorithm for Deep RL



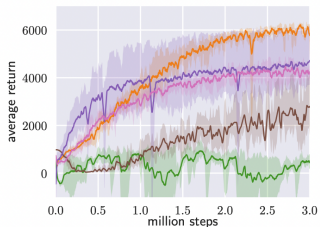
(a) Hopper-v1



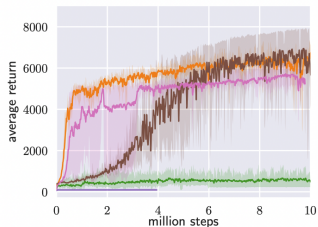
(b) Walker2d-v1



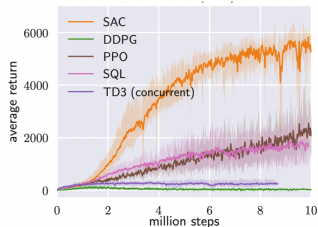
(c) HalfCheetah-v1



(d) Ant-v1



(e) Humanoid-v1



(f) Humanoid (rllab)

Question and Answering (Q&A)



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen