Policy Gradient

Policy- Based : (directly parametrise the policy)

a. π_θ(s, a) =P $[a|s,\theta]$ ⟶
- Softmax function
- Gaussion Distribution
- Neural network

b. Three kinds of objective Functions

$$\begin{cases} J_1(\theta) = V^{\pi_\theta}(s_1) = E_{\pi_\theta}[v_1] & \text{In episodic environments using the start value} \\ J_{avV}(\theta) = \sum_s d^{\pi_\theta}(s) V^{\pi_\theta}(s) & \text{In continuing environments using the average value} \\ J_{avR}(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(\text{s, a}) R_s^a & \text{Average reward per time-step} \end{cases}$$

where : $d^{\pi_\theta}(s)$ is stationary distribution of Markov chain for π_θ

An optimisation problem

How to solve: (using gradient)

a. Policy Gradient: Policy gradient algorithms search for a local maximum in $J(\theta)$ by ascending the gradient of the policy, w.r.t. parameters θ

$$\Delta\theta = \alpha\nabla_\theta J(\theta) = \alpha\begin{pmatrix}\frac{\partial J(\theta)}{\partial\theta_1}\\ \cdots \\ \frac{\partial J(\theta)}{\partial\theta_n}\end{pmatrix}$$

→ **It means adjusting θ in the direction of gradient**

b. Computing Gradients

Numerical method: Finite Differences

For each dimension k ∈ [1, n]:

$$\frac{\partial J(\theta)}{\partial\theta_k} \approx \frac{J(\theta + \epsilon u_k) - J(\theta)}{\epsilon}$$

where u_k is unit vector with 1 in kth component, 0 elsewhere

Analysis method: Monte-Carlo **Policy Gradient (Reinforce)**

$$\theta_{t+1} = \theta_t + \alpha\nabla_\theta log\pi_\theta(s, a)\, v_t$$

Where does MC Policy Gradient come from:

Likelihood ratios： :

Math trick: Likelihood ratios exploit the following identity

$$\nabla_\theta \pi_\theta(s, a) = \pi_\theta(s, a) \frac{\nabla_\theta \pi_\theta(s, a)}{\pi_\theta(s, a)}$$
$$= \pi_\theta(s, a) \nabla_\theta log \pi_\theta(s, a) \quad \text{(based on: } dlog(y) = dy/y)$$

The score function is $\nabla_\theta log \pi_\theta(s, a)$

For example:

Softmax policy:

$$\pi_\theta(s, a) \propto e^{\phi(s,a)^T \theta}$$

The score function is $\nabla_\theta log \pi_\theta(s, a) = \phi(s,a) - E_{\pi_\theta}[\phi(s, \cdot)]$

Gaussian policy:

$$\pi_\theta(s, a) \propto \mathcal{N}(\mu(s), \sigma^2)$$

The score function is $\nabla_\theta log \pi_\theta(s, a) = \frac{(a - \mu(s))\phi(s)}{\sigma^2}$

Policy Gradient Theorem:

Theorem

For any differentiable policy $\pi_\theta(s, a)$,
for any of the policy objective functions $J = J_1, J_{avR}$, *or* $\frac{1}{1-\gamma} J_{avV}$,
the policy gradient is

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

Substitute Sample for Expection:

a. Updating parameters by stochastic gradient ascent

b. Using return v_t as an unbiased sample of $Q^{\pi_\theta}(s_t, a_t)$

$$\Delta\theta_t = \alpha \nabla_\theta log \pi_\theta(\text{s, a}) \, v_t$$

or updating form:

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta log \pi_\theta(\text{s, a}) \, v_t$$

Reinforce with Baseline B

1. Why baseline?

-In reinforce, we substitute Sample for Expection, therefore there will be variance.

$\Rightarrow \quad \nabla_\theta \mathrm{J}(\theta) = E_{\pi_\theta}[\nabla_\theta log\pi_\theta(\mathrm{s, a})\,(Q^{\pi_\theta}(s,a) - B)]$

$\Downarrow$

$E_{\pi_\theta}[\nabla_\theta log\pi_\theta(\mathrm{s, a})\,B] = 0$

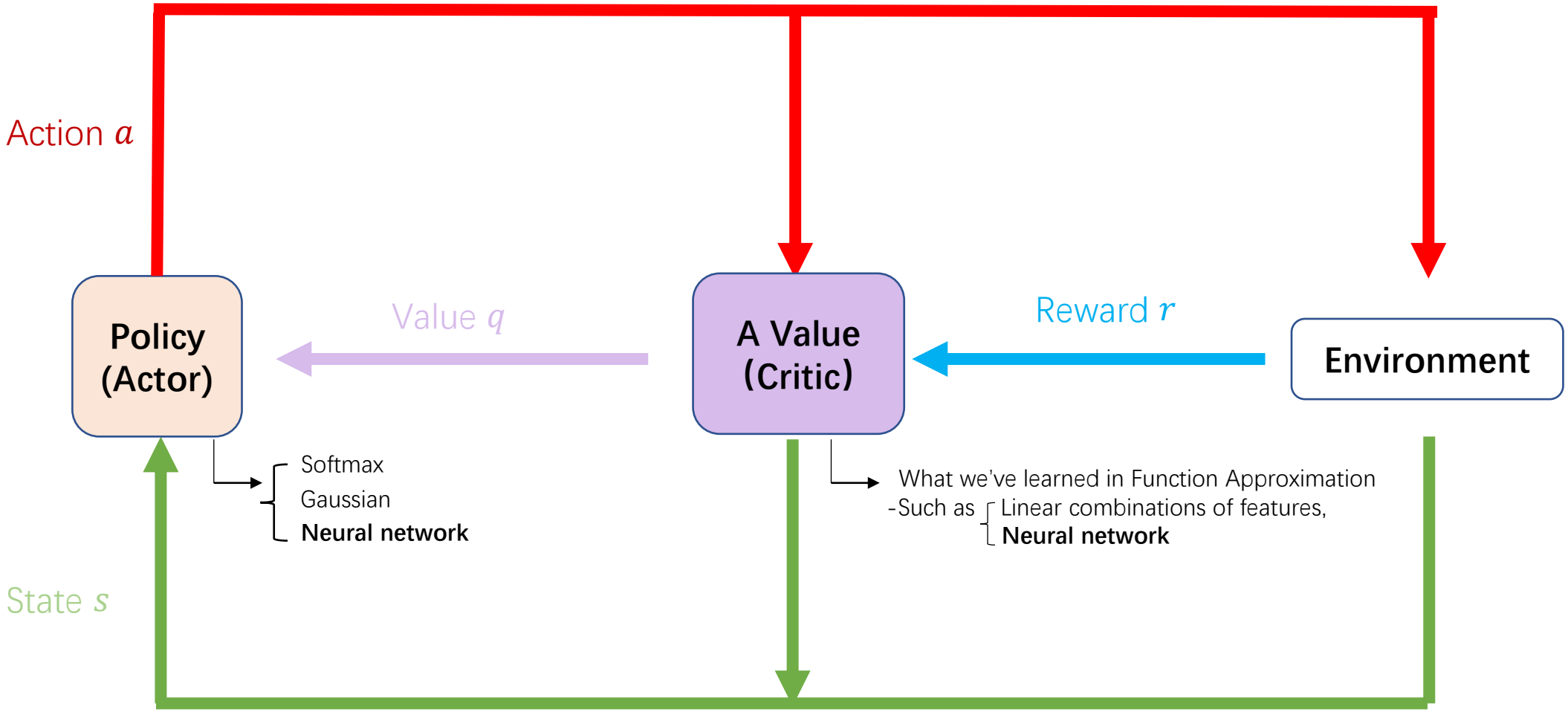
$\Rightarrow \quad \theta_{t+1} = \theta_t + \alpha(v_t - B)\nabla_\theta log\pi_\theta(\mathrm{s, a})\,v_t$

Expection doesn't change, but the variance decreases

2.Choices of Baseline:

- B = 0
- B = $V^{\pi_\theta}(s)$

Actor-Critic



Fundamental idea:

We use a critic to estimate the action-value function,

$$Q_w(\text{s, a}) \approx Q^{\pi_\theta}(s, a)$$

Actor-critic algorithms maintain two sets of parameters:

Critic Updates action-value function parameters w

Actor Updates policy parameters θ, in direction suggested by critic

$$\Rightarrow \quad \Delta\theta = \alpha \nabla_\theta log \pi_\theta(\text{s, a}) Q_w(\text{s, a})$$

Actor-Critic Algorithms with Baseline:

$$\begin{cases} \text{B} = V^{\pi_\theta}(s) \\ A^{\pi_\theta}(s, a) = Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s) \longrightarrow \text{Advantage Function} \\ A(s, a) = Q_w(\text{s, a}) - V_v(\text{s}) \end{cases}$$

$$\Rightarrow \quad \nabla_\theta \text{J}(\theta) = E_{\pi_\theta}[\nabla_\theta log \pi_\theta(\text{s, a}) A(s, a)]$$

So what we should do is estimating the Advantage Function

Estimating the Advantage Function

For the true value function $V^{\pi_\theta}(s)$, the TD error $\boldsymbol{\delta^{\pi_\theta}}$

$$\delta^{\pi_\theta} = r + \gamma V^{\pi_\theta}(s') - V^{\pi_\theta}(s)$$

is an unbiased estimate of the advantage function

$\Rightarrow$ In practice we can use an approximate TD error

$$\delta_v = r + \gamma V_v(s') - V_v(s)$$

This approach only requires one set of critic parameters v

$$\Rightarrow \quad \nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta log \pi_\theta(s, a)\delta_v]$$

So , we can use different Time-Scales "TD Target" in actor and critic

For actor:

- The policy gradient can be estimated at many time-scales

MC

$$\Delta\theta_t = \alpha(v_t - V_v(s_t))\ \nabla_\theta log\pi_\theta(\text{s, a})$$

one-step TD

$$\Delta\theta_t = \alpha\left(r + \gamma V_v(s_{t+1}) - V_v\ (s_t)\right)\nabla_\theta log\pi_\theta(\text{s, a})$$

forward-view TD(λ)

$$\Delta\theta_t = \alpha\left(v_t^\lambda - V_v\ (s_t)\right)\ \nabla_\theta log\pi_\theta(\text{s, a})$$

backward-view TD(λ)

$$\delta = r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t)$$

$$e_{t+1} = \lambda e_t + \nabla_\theta log\pi_\theta(\text{s, a})$$

$$\Delta\theta_t = \alpha\delta e_t$$

For critic:

- Value function $V_\theta(s)$ can also be estimated by many targets at different time-scales

MC

$$\Delta\theta = \alpha(v_t - V_\theta(s))\phi(s)$$

TD(0)

$$\Delta\theta = \alpha(r + \gamma V(s') - V_\theta(s))\phi(s)$$

forward-view TD(λ)

$$\Delta\theta = \alpha(v_t^\lambda - V_\theta(s))\phi(s)$$

backward-view TD(λ)

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$

$$e_t = \gamma\lambda e_{t-1} + \phi(s_t)$$

$$\Delta\theta = \alpha\delta_t e_t$$

They are what we've learned in Function Approximator

Summary

In practice:

Agent is controlled by policy network

An other neural networks is used to do function approximation

For Training:

Update the policy network (actor) by policy gradient.

Update the value network (critic) by TD learning.

1. Observe the state s_t; randomly sample action a_t according to $\pi(\cdot\,|s_t;\boldsymbol{\theta}_t)$.
2. Perform a_t; observe new state s_{t+1} and reward r_t.
3. Randomly sample a_{t+1} according to $\pi(\cdot\,|s_{t+1};\boldsymbol{\theta}_t)$. (Do not perform a_{t+1}.)
4. Evaluate value network: $q_t = q(s_t, a_t; \mathbf{w}_t)$ and $q_{t+1} = q(s_{t+1}, a_{t+1}; \mathbf{w}_t)$.
5. Compute the TD error: $\delta_t = q_t - (r_t + \gamma \cdot q_{t+1})$.
6. Differentiate value network: $\mathbf{d}_{w,t} = \frac{\partial q(s_t, a_t; \mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}=\mathbf{w}_t}$.
7. Update value network: $\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \cdot \delta_t \cdot \mathbf{d}_{w,t}$.
8. Differentiate policy network: $\mathbf{d}_{\theta,t} = \frac{\partial \log \pi(a_t|s_t,\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t}$.
9. Update policy network: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \beta \cdot \delta_t \cdot \mathbf{d}_{\theta,t}$.

Baseline

- The policy gradient has many equivalent forms

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ v_t\right] && \text{REINFORCE} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ Q^w(s,a)\right] && \text{Q Actor-Critic} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ A^w(s,a)\right] && \text{Advantage Actor-Critic} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ \delta\right] && \text{TD Actor-Critic} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ \delta e\right] && \text{TD}(\lambda)\text{ Actor-Critic} \\
G_\theta^{-1}\nabla_\theta J(\theta) &= w && \text{Natural Actor-Critic}
\end{aligned}
$$

- 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 $V^\pi(s)$