Robotic Navigation and Exploration

Week 13: Reinforcement Learning (II)

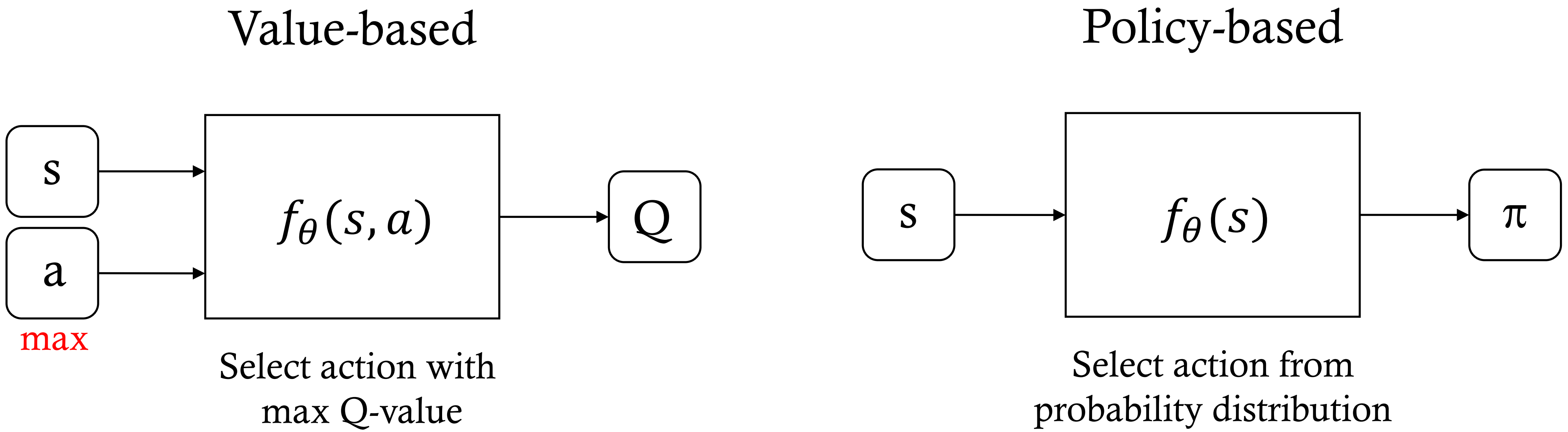
Min-Chun Hu anitahu@cs.nthu.edu.tw
CS, NTHU

Outline

- Policy-based Reinforcement Learning
  - Policy Gradient and Actor Critic
  - Sampling Efficiency Problem (TRPO & PPO)
  - Deep Deterministic Policy Gradient (DDPG) & Soft Actor-Critic (SAC)
- Lab6: Model free RL for Mapless Navigation

Policy-based vs. Value-based Method

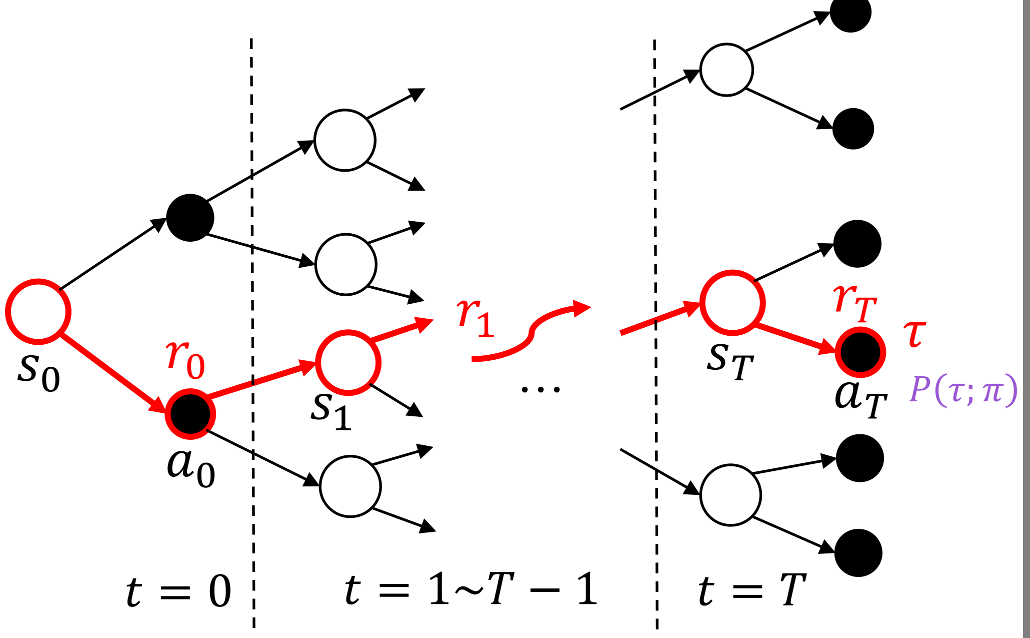
- Value-based method aims to learn the action-value function and policy-based method aims to learn the policy function.



Policy Gradient

- For policy-based method, we construct a function approximator to learn the mapping from state to the action.
- If the computation of gradient for the parameters is achievable, we can utilize iterative optimization method to optimize the policy.
- Definition
  - Trajectory $\tau = \{s_0, a_0, s_1, a_1, \dots, s_T, a_T\}$
  - Return $G(\tau) = r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots + \gamma^T r_t$
  - Probability of a trajectory

$$P(\tau;\pi) = P(s_0)\prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t)$$



Policy Gradient

- Gradient of the policy

$$\nabla_\theta J(\theta) = \nabla_\theta E\left[\sum_{t=0}^{T} \gamma^t r_t\right]$$

$$= \nabla_\theta \sum_\tau P\big(\tau; \pi_\theta(a|s)\big) * G(\tau)$$

$$= \sum_\tau \nabla_\theta P(\tau; \pi_\theta) * G(\tau)$$

$$= \sum_\tau P(\tau; \pi_\theta) \frac{\nabla_\theta P(\tau; \pi_\theta)}{P(\tau; \pi_\theta)} G(\tau)$$

$$= \sum_\tau P(\tau; \pi_\theta) \nabla_\theta \log P(\tau; \pi_\theta) * G(\tau)$$

$$= \mathbb{E}_\tau[\nabla_\theta \log P(\tau; \pi_\theta) * G(\tau)]$$

- Probability of a trajectory

$$\nabla_\theta \log P(\tau; \pi_\theta)$$

$$= \nabla_\theta \log P(s_0) \prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t)$$

$$= \nabla_\theta \sum_{t=0}^{T-1} \log P(s_{t+1}|s_t, a_t) + \nabla_\theta \sum_{t=0}^{T-1} \log \pi_\theta(a_t|s_t)$$

$$= \nabla_\theta \sum_{t=0}^{T-1} \log \pi_\theta(a_t|s_t)$$

Policy Gradient:

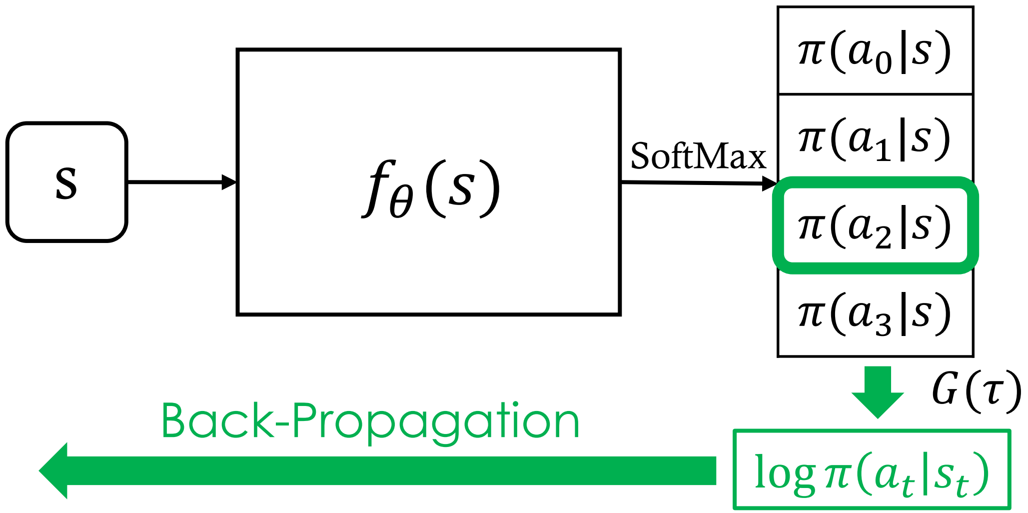
$$\nabla_\theta J(\theta) = \mathbb{E}_\tau\left[\nabla_\theta \sum_{t=0}^{T-1} log\, \pi_\theta(a_t|s_t) * G(\tau)\right]$$

Policy Gradient

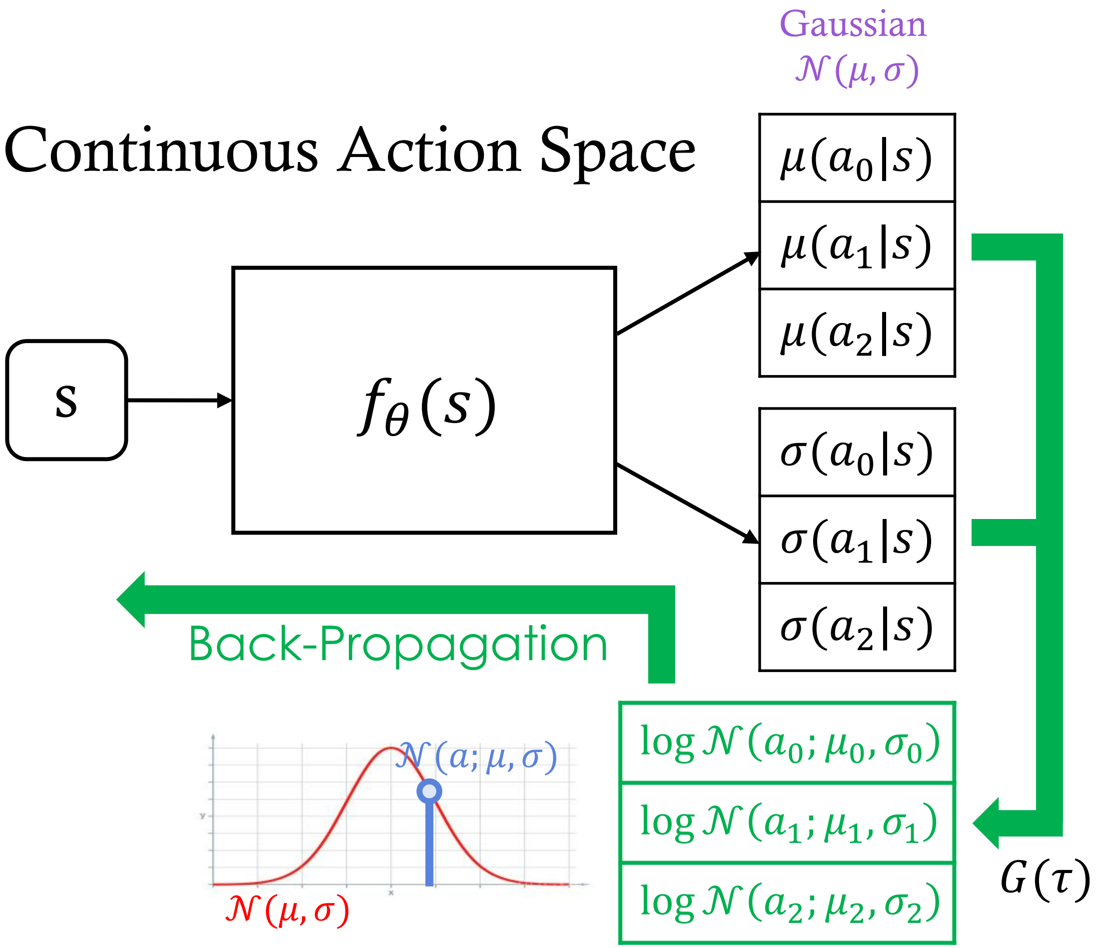
Policy Gradient:

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau \left[\nabla_\theta \sum_{t=0}^{T-1} \boldsymbol{\log \pi_\theta(a_t|s_t)} * G(\tau) \right]$$

Discrete Action Space

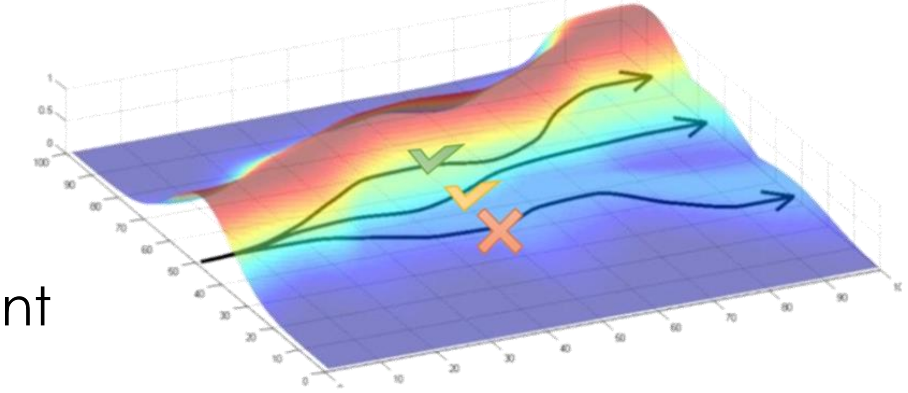


Continuous Action Space



REINFORCE Algorithm

- Monte-Carlo estimation of the policy gradient



$$s_0 \;\text{—}\; a_0 \;\overset{r_0}{\text{——}}\; s_1 \;\text{—}\; a_1 \;\overset{r_1}{\text{——}}\; s_2 \;\text{—}\; a_2 \;\overset{r_2}{\text{——}}\; s_3 \; \text{- - - - -} \; s_{T-1} \;\text{—}\; \bullet \;\overset{r_{T-1}}{\text{——}}\; \boxed{s_T}$$

$$\log \pi_\theta(a_0|s_0) \longrightarrow G_0$$

$$\log \pi_\theta(a_1|s_1) \longrightarrow G_1$$

$$\log \pi_\theta(a_2|s_2) \longrightarrow G_2$$

$$\log \pi_\theta(a_{T-1}|s_{T-1}) \longrightarrow G_{T-1}$$

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau \left[\nabla_\theta \sum_{t=0}^{T-1} \log \pi_\theta(a_t|s_t) * G_t(\tau) \right] \approx \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta\left(a_t^i \middle| s_t^i\right) * G_t\left(\tau^i\right)$$

(Episode Update)

Baseline for Policy Gradient

- Adding an appropriate baseline function can reduce the variance of the estimation for policy gradient.

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau[\nabla_\theta \log \pi_\theta(a_t|s_t) * G_t(\tau)] = \mathbb{E}_\tau[\nabla_\theta \log \pi_\theta(a_t|s_t) * (G_t(\tau) - \underbrace{b(s_t)}_{\text{Baseline}})]$$

$$\Rightarrow \mathbb{E}_\tau[\nabla_\theta \log \pi_\theta(a_t|s_t) * b(s_t)] = 0$$

Proof:

$$\mathbb{E}_\tau[\nabla_\theta \log \pi_\theta(a_t|s_t) * b(s_t)]$$
$$= \mathbb{E}_{s_{0:t},a_{0:t}}\left[\mathbb{E}_{s_{t+1:T},a_{t+1:T}}[\nabla_\theta \log \pi_\theta(a_t|s_t) * b(s_t)]\right]$$
$$= \mathbb{E}_{s_{0:t},a_{0:t}}\left[b(s_t) * \mathbb{E}_{s_{t+1:T},a_{t+1:T}}[\nabla_\theta \log \pi_\theta(a_t|s_t)]\right]$$
$$= \mathbb{E}_{s_{0:t},a_{0:t}}\left[b(s_t) * \sum_{a_t \in A} \sum_{s_{t+1} \in S} \dots \sum_{s_T \in S} \pi_\theta(a_t, s_t) P(s_{t+1}|s_t, a_t) \dots P(s_T|s_{T-1}, a_{T-1})(\nabla_\theta \log \pi_\theta(a_t|s_t))\right]$$
$$= \mathbb{E}_{s_{0:t},a_{0:t}}\left[b(s_t) * \sum_{a_t \in A} \pi_\theta(a_t|s_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{s_{t+1} \in S} P(s_{t+1}|s_t, a_t) \sum_{a_{t+1} \in A} \dots \sum_{s_T \in S} P(s_T|s_{T-1}, a_{T-1})\right]$$
$$= \mathbb{E}_{s_{0:t},a_{0:t}}\left[b(s_t) * \mathbb{E}_{a_t}[\nabla_\theta \log \pi_\theta(a_t|s_t)]\right] = \mathbb{E}_{s_{0:t},a_{0:t}}[b(s_t) * 0] = 0$$

$$\mathbb{E}_{a_t}[\nabla_\theta \, log \, \pi_\theta(a_t|s_t)]$$

$$= \sum_{a_t \in A} \pi_\theta(a_t|s_t) \nabla_\theta \, log \, \pi_\theta(a_t|s_t)$$

$$= \sum_{a_t \in A} \pi_\theta(a_t|s_t) \frac{\nabla_\theta \pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)}$$

$$= \sum_{a_t \in A} \nabla_\theta \pi_\theta(a_t|s_t)$$

$$= \nabla_\theta \sum_{a_t \in A} \pi_\theta(a_t|s_t)$$

$$= \nabla_\theta 1$$

$$= 0$$

Baseline for Policy Gradient

- Consider the variance of policy gradient

$$Var\left(\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) * \left(G_t(\tau) - b(s_t)\right)\right)$$
$$\approx \sum_{t=0}^{T-1} \mathbb{E}_\tau\left[\left(\nabla_\theta \log \pi_\theta(a_t|s_t) * \left(G_t(\tau) - b(s_t)\right)\right)^2\right]$$
$$\approx \sum_{t=0}^{T-1} \mathbb{E}_\tau[(\nabla_\theta \log \pi_\theta(a_t|s_t))^2] * \mathbb{E}_\tau[\left(\left(G_t(\tau) - b(s_t)\right)\right)^2]$$

Setting $b(s_t) \approx \mathbb{E}_\tau[G_t(\tau)]$ to approximate the expected return will have low variance.

Off-Policy with Importance Sampling

- We compute the policy gradient of π_θ by collecting the data under the probability of policy π_θ. Once we update the policy, we need to collect the data again.

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p_\theta(\tau)}[G(\tau)\nabla_\theta \log p_\theta(\tau)]$$

- If we want to reuse the data sampled before updating the policy, we can apply the importance sampling technique to estimate the policy gradient.

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}\left[\frac{p_\theta(\tau)}{p_{\theta'}(\tau)} G(\tau)\nabla_\theta \log p_\theta(\tau)\right]$$

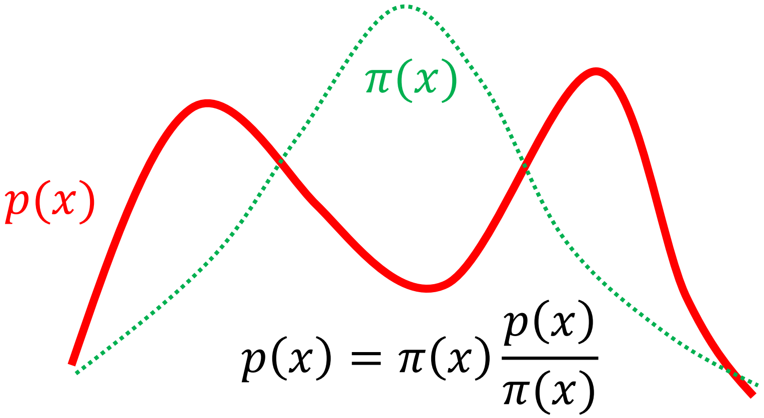
Sample from $p_{\theta'}(\tau)$

Importance Weight

$$\boxed{\mathbb{E}_{x\sim p}[f(x)] = \mathbb{E}_{x\sim q}\left[\frac{p(x)}{q(x)} f(x)\right]}$$

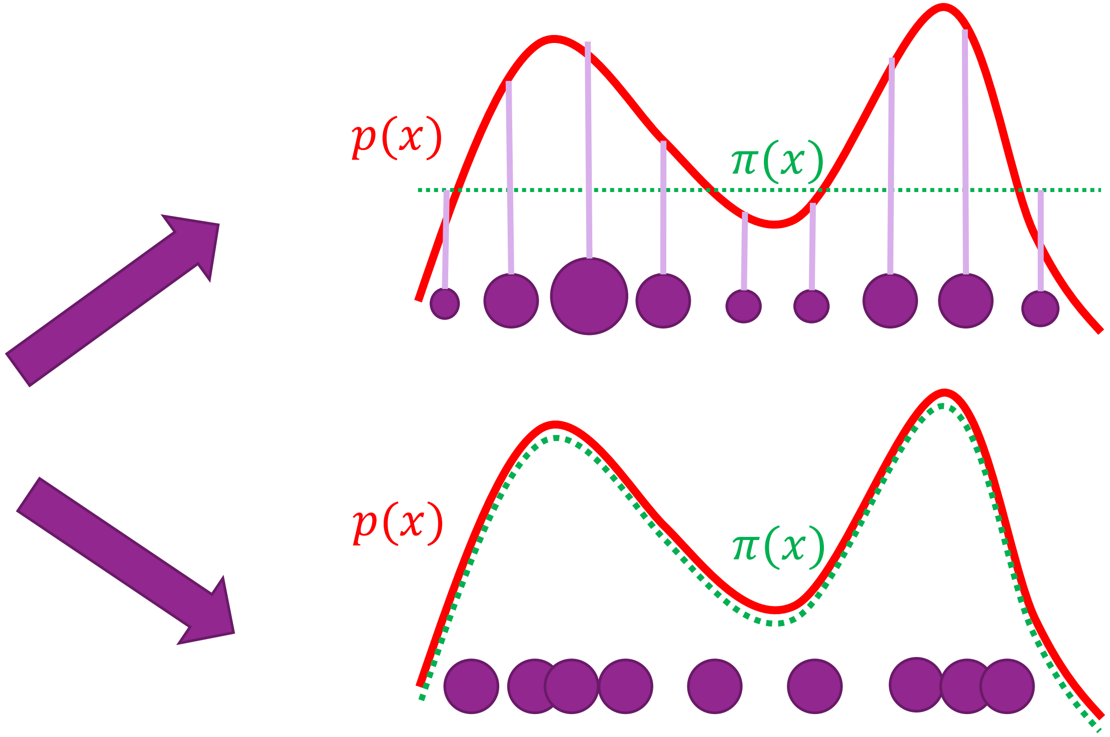
Importance Sampling Review

- Important sampling adopts discrete multinomial to approximate arbitrary distribution. More sampling particles will have more accurate approximation.



$$p(x) = \pi(x)\frac{p(x)}{\pi(x)}$$

1. *Sampling* x_i *from* $\pi(x)$
2. *Calculate* $w_i = \frac{p(x_i)}{\pi(x_i)}$
3. *Sampling* x *from* $mul(x_i, w_i)$



Off-Policy with Importance Sampling

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p_\theta(\tau)}[G(\tau)\nabla_\theta \log p_\theta(\tau)]$$

$$= \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}\left[\frac{p_\theta(\tau)}{p_{\theta'}(\tau)} G^{\theta'}(\tau)\nabla_\theta \log p_\theta(\tau)\right]$$

$$= \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}\left[\frac{p_\theta(s_t,a_t)}{p_{\theta'}(s_t,a_t)} G^{\theta'}(\tau)\nabla_\theta \log p_\theta(s_t,a_t)\right]$$

$$= \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}\left[\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)} \cancel{\frac{p_\theta(s_t)}{p_{\theta'}(s_t)}} G^{\theta'}(\tau)\nabla_\theta \log p_\theta(s_t,a_t)\right]$$

$$J^{\theta'}(\theta) = \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}[\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)} G^{\theta'}(\tau)]$$

Trust Region Policy Optimization (TRPO)

$$J^{\theta'}_{TRPO}(\theta) = \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}[\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)} G^{\theta'}(\tau)]$$

Subject to KL$(\theta, \theta')<\delta$

Proximal Policy Optimization (PPO)

$$\nabla^{PPO}_{\theta'} J(\theta) = J^{\theta'}(\theta) - \beta KL(\theta,\theta')$$

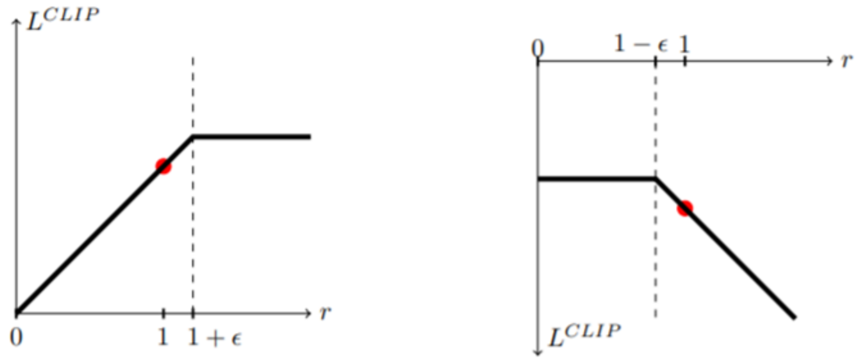
$$J^{\theta'}(\theta) = \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}[\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)} G^{\theta'}(\tau)]$$

Proximal Policy Optimization

Proximal Policy Optimization (PPO)

$$\nabla_{\theta'}^{PPO} J(\theta) = J^{\theta'}(\theta) - \beta KL(\theta, \theta')$$

$$J^{\theta'}(\theta) = \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}[\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)} G^{\theta'}(\tau)]$$



Proximal Policy Optimization (Clip Version)

$$\nabla_{\theta'}^{PPO2} J(\theta) \approx \sum_{s_t, a_t} min\left(\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)} G^{\theta'}(\tau), clip\left(\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon\right) G^{\theta'}(\tau)\right)$$

Actor-Critic

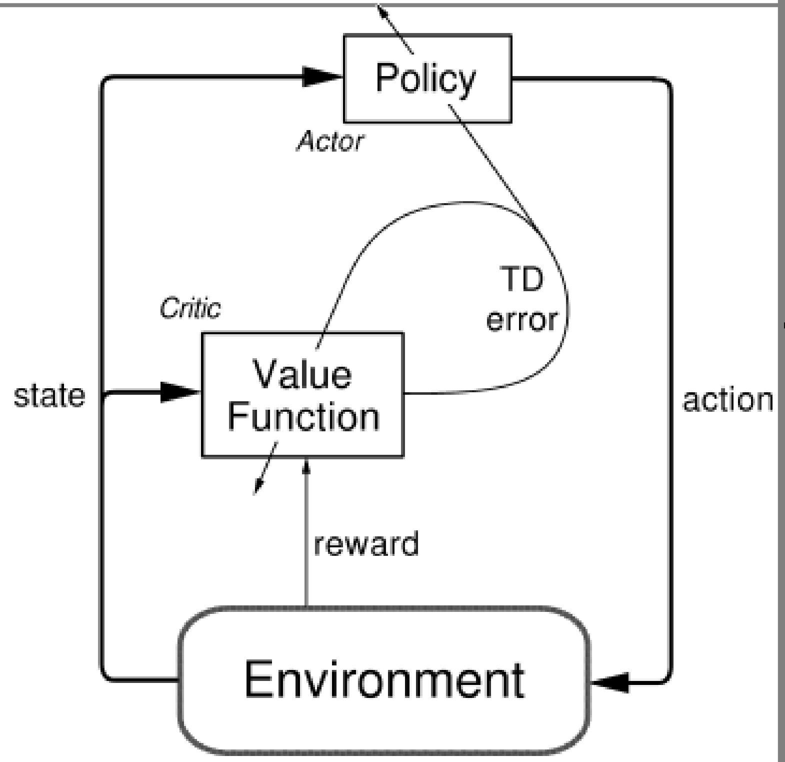
$$\nabla_\theta J(\theta) = \mathrm{E}_\pi\left[\nabla_\theta \sum_{t=0}^{T-1} \log \pi_\theta(a_t|s_t)\, G_t(\tau)\right]$$ Policy Gradient

$$= \mathrm{E}_\pi\left[\nabla_\theta \sum_{t=0}^{T-1} \log \pi_\theta(a_t|s_t)\, Q(s_t, a_t)\right]$$ Actor-Critic

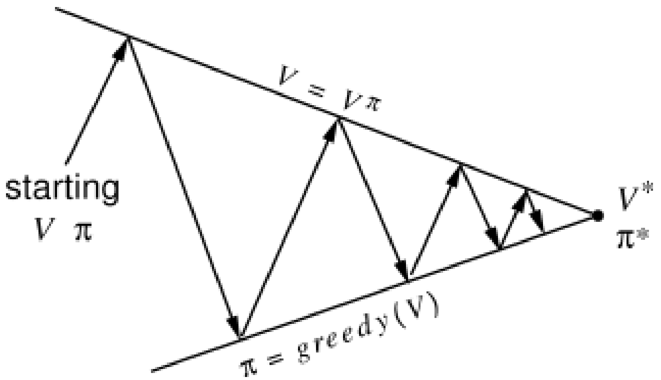
$$= \mathrm{E}_\pi\left[\nabla_\theta \sum_{t=0}^{T-1} \log \pi_\theta(a_t|s_t)\, [Q(s_t, a_t) - B(s_t)]\right]$$ Actor-Critic with Baseline

$$= \mathrm{E}_\pi\left[\nabla_\theta \sum_{t=0}^{T-1} \log \pi_\theta(a_t|s_t)\, [Q(s_t, a_t) - V(s_t)]\right]$$ Advantage Actor-Critic

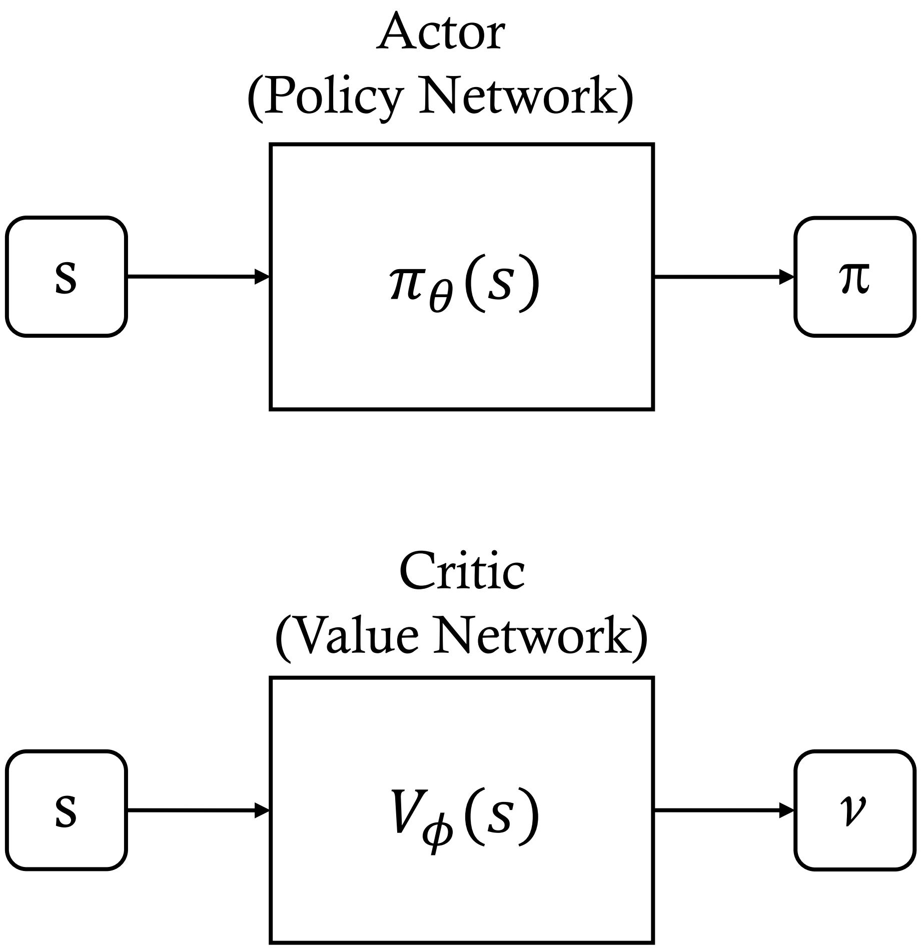
$$= \mathrm{E}_\pi\left[\nabla_\theta \sum_{t=0}^{T-1} \log \pi_\theta(a_t|s_t)\, [r_t + \gamma V(s_{t+1}) - V(s_t)]\right]$$ TD Actor-Critic



Policy Iteration



TD Actor-Critic



Actor (Policy) Loss:

$$L(\theta) = \mathbb{E}[-\log \pi_\theta(a|s)\,\underbrace{\left(r + \gamma V(s') - V(s)\right)}_{\text{Advantage}} - \underbrace{\beta H\left(\pi_\theta(a|s)\right)}_{\text{Entropy Term (Encourage Exploration)}}]$$

Critic (Value) Loss:

$$L(\phi) = \mathbb{E}[\frac{1}{2}\underbrace{\left(r + \gamma V(s') - V(s)\right)}_{\text{TD-Error}}^2]$$

Deterministic Policy Gradient (DPG)

- The DPG algorithm maintains an actor function ***μ(s|θ)*** which specifies the current policy by deterministically mapping states to a specific action.
  - **Stochastic Policy**

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim P}[r(s_t, a_t) + \gamma \boxed{\mathbb{E}_{a+1 \sim \pi}}[Q^{\pi}(s_{t+1}, a_{t+1})]]$$

  - **Deterministic Policy**

$$Q^{\mu}(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim P}[r(s_t, a_t) + \gamma Q^{\mu}(s_{t+1}, \boxed{\mu(s_{t+1})})]$$

- Without the stochastic property of the policy distribution, the policy gradient can be estimated simpler by chain rules.

$$\nabla_{\theta} J(\mu_{\theta}) \approx \int_{s} P^{\mu}(s)\, \nabla_{\theta} \mu_{\theta}(s) \nabla_a Q^{\mu}(s, a) \Big|_{a=\mu_{\theta}(s)} ds$$
$$= \mathbb{E}_{s \sim P^{\mu}}[\nabla_{\theta} \mu_{\theta}(s) \nabla_a Q^{\mu}(s, a)|_{a=\mu_{\theta}(s)}]$$

On-policy and Off-policy DPG

- On-policy DPG

$$\nabla_\theta J(\mu_\theta) = \int_S P^\mu(s)\, \nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a) ds$$
$$= \mathbb{E}_{s\sim P^\mu}[\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)}]$$

Update Parameters

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

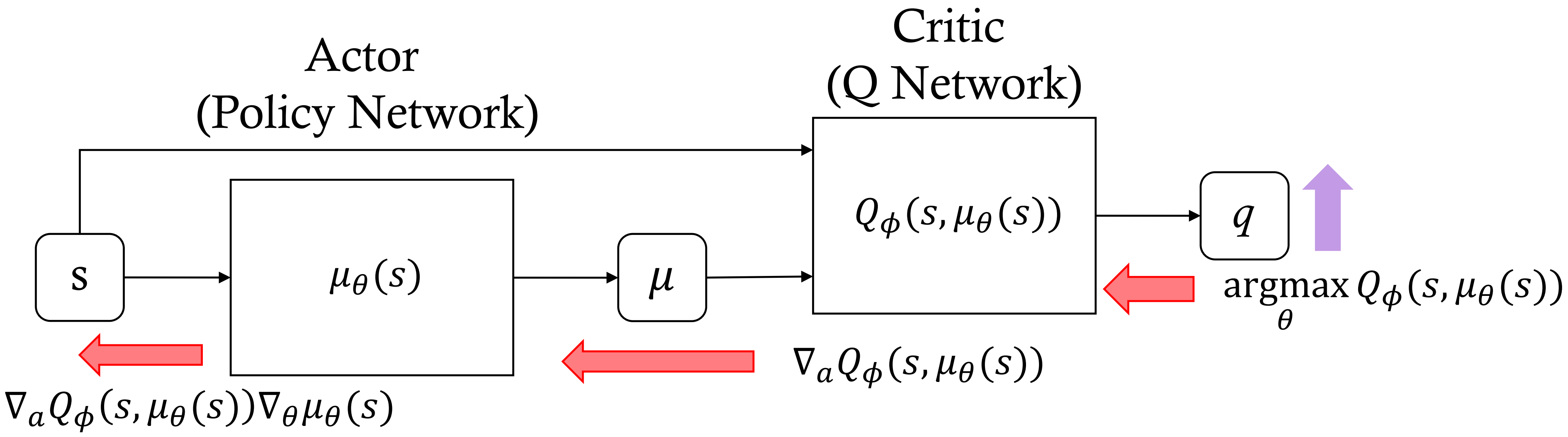
- Off-policy DPG

$$\nabla_\theta J_\beta(\mu_\theta) \approx \int_S P^\beta(s)\, \nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a) ds$$
$$= \mathbb{E}_{s\sim P^\beta}[\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)}]$$

Assume $\underset{a}{\operatorname{argmax}}\, Q(s,a) = \mu_\theta(s_{t+1})$

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

Deterministic Policy Gradient (DPG)



Actor (Policy) Loss:

$L(\theta) = \mathbb{E}[-Q_\phi(s, \mu_\theta(s))]$

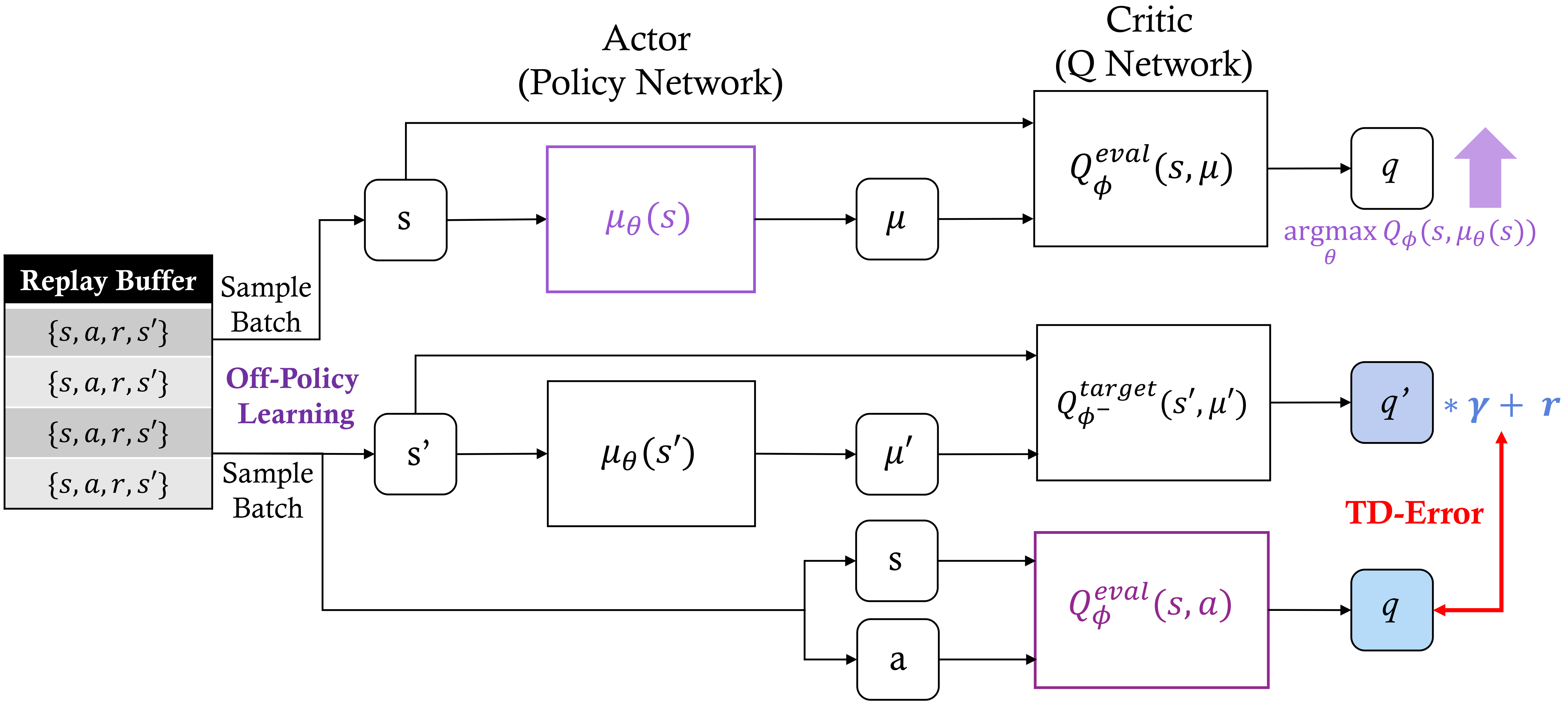
Critic (Q) Loss:

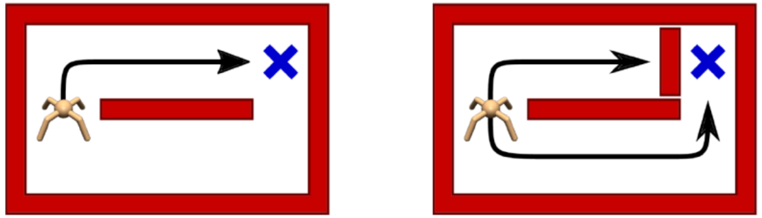
$L(\phi) = \mathbb{E}[\frac{1}{2}\left(r + \gamma Q_\phi(s', \mu(s')) - Q(s,a)\right)^2]$

TD-Error

Off-Policy Learning

Deep Deterministic Policy Gradient (DDPG)





Exploration Issue

- The formulation of classic RL framework implies an optimal deterministic policy, while it conflict to the stochastic policy that encourages the exploration of the environment.

- We usually add an regularized entropy term to maintain the stochastic property in stochastic policy algorithm such as policy gradient or actor-critic.

$$L_{actor}(\theta) = \mathbb{E}[-\log \pi_\theta(a|s)\, A(s) - \beta H\big(\pi_\theta(a|s)\big)]$$

- An extension of classic RL framework is **Maximum Entropy Reinforcement Learning Framework**, which take the long-term entropy of the policy distribution into consideration to ensure the exploration of the environment.

$$\pi^*_{MaxEnt} = \underset{\pi}{\operatorname{argmax}} \sum_t \mathbb{E}_\pi \left[r(s_t, a_t) + \alpha \mathcal{H}\big(\pi(.|s_t)\big)\right]$$

Consider the future entropy

Maximum Entropy RL Framework

$$\pi^*_{MaxEnt} = \underset{\pi}{\operatorname{argmax}} \sum_t \mathbb{E}_\pi \left[r(s_t, a_t) + \alpha \mathcal{H}\big(\pi(.|s_t)\big)\right]$$

Maximum Entropy Objective

$$V_{soft}(s_t) = \mathbb{E}_\pi[Q_{soft}(s_t, a_t) - \alpha \log \pi(a_t|s_t)]$$

Energy-based Policy

Soft Value Function

$$\varepsilon(s_t, a_t) = -\frac{1}{\alpha} Q_{soft}(s_t, a_t)$$

$$\pi(a_t|s_t) \propto \exp(Q_{soft}(s_t, a_t))$$

$$V_{soft}(s_t) = \log \int \exp\left(\frac{1}{\alpha} Q_{soft}(s_t, a)\right) da$$

$$= \operatorname{soft}\max_a Q_{soft}(s_t, a)$$

Soft Actor-Critic

- Soft Actor Critic (SAC) algorithm is an off-policy algorithm based on the soft policy iteration of maximum entropy RL framework.
- SAC can be seen as the stochastic extension of the DDPG, which adopt the **reparameterization trick** to estimate the gradient of gaussian policy.

Actor (Policy) Loss:

$$L(\theta) = \mathbb{E}[\alpha \log \pi_\theta(f_\theta(\epsilon_t; s_t)|s_t) - Q_\phi(s_t, f_\theta(\epsilon_t; s_t))]$$

$$\boxed{a_t = f_\theta(\epsilon_t; s_t)\ (Reparameterize)}$$

Critic (Q) Loss:

$$L(\phi) = \mathbb{E}\left[\frac{1}{2}\left(-\gamma\alpha \log \pi(.|s_{t+1}) + r(s_t, a_t) + \gamma Q_\phi\left(s_{t+1}, f_\theta(\epsilon_{t+1}; s_{t+1})\right) - Q_\phi(s_t, a_t)\right)^2\right]$$

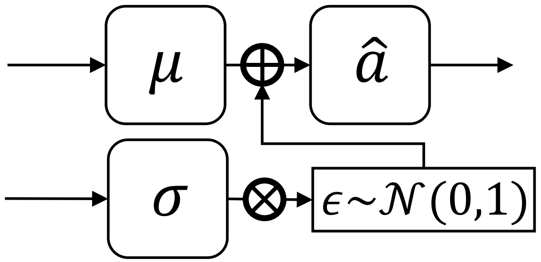
Actor Loss (DDPG):

$$L(\theta) = \mathbb{E}[-Q_\phi(s, \mu_\theta(s))]$$

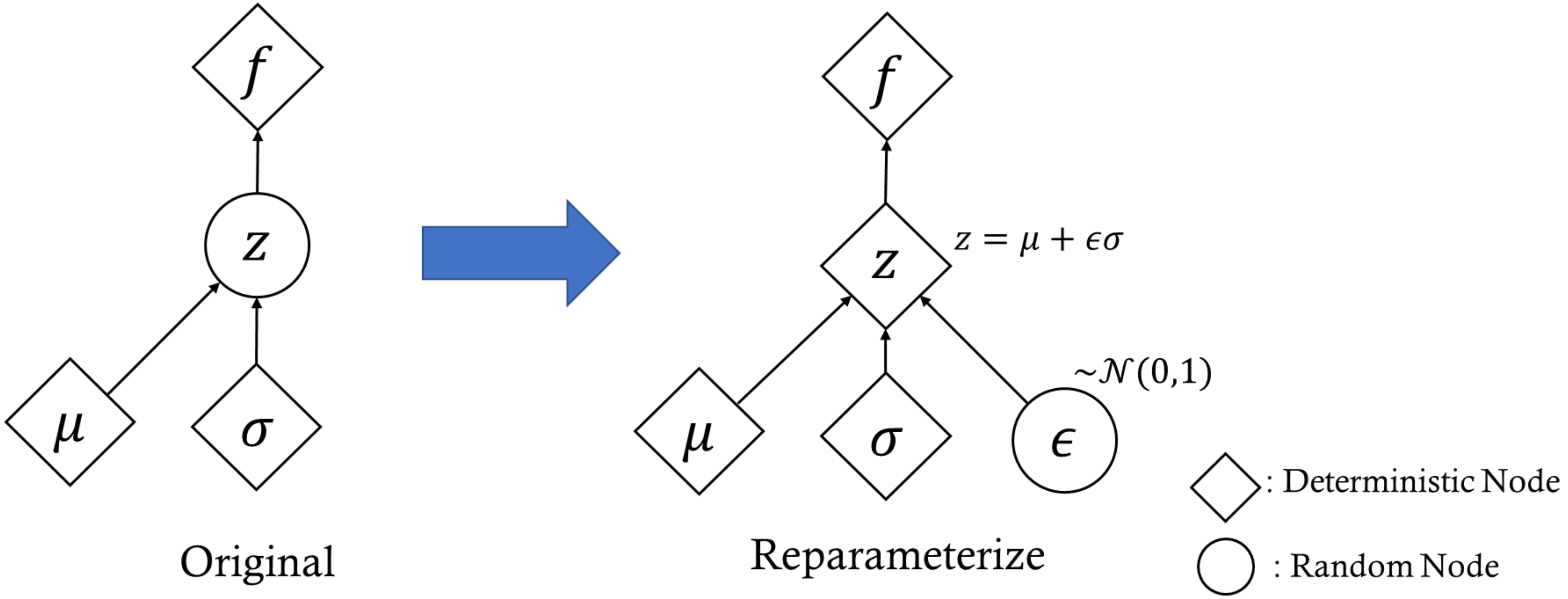
Critic Loss (DDPG):

$$L(\phi) = \mathbb{E}[\tfrac{1}{2}\left(r + \gamma Q_\phi(s', \mu(s')) - Q(s,a)\right)^2]$$

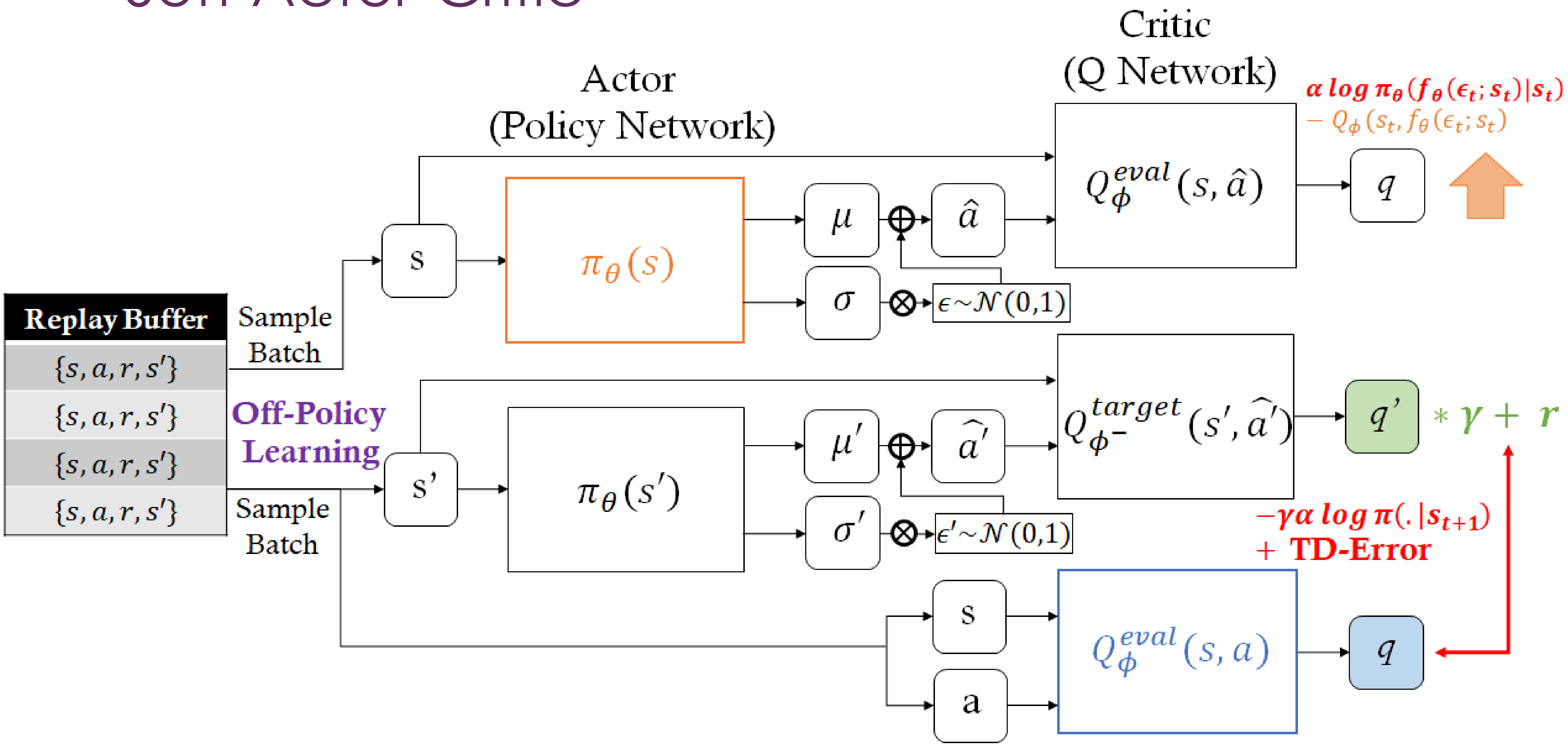
Reparameterization



- Unbias gradient estimator for Gaussian sampling



Soft Actor-Critic



Reinforcement Learning Algorithms