

# REINFORCEMENT LEARNING & ADVANCED DEEP

## M2 DAC

### TME 7. Continuous Actions

Ce TME a pour objectif d'expérimenter l'approche DDPG pour environnements à actions continues.

#### DDPG

Implémenter l'algorithme DDPG suivant:

---

**Algorithm 1** Deep Deterministic Policy Gradient

---

- 1: Input: initial policy parameters  $\theta$ , Q-function parameters  $\phi$ , empty replay buffer  $\mathcal{D}$
  - 2: Set target parameters equal to main parameters  $\theta_{\text{targ}} \leftarrow \theta$ ,  $\phi_{\text{targ}} \leftarrow \phi$
  - 3: **repeat**
  - 4:   Observe state  $s$  and select action  $a = \text{clip}(\mu_{\theta}(s) + \epsilon, a_{\text{Low}}, a_{\text{High}})$ , where  $\epsilon \sim \mathcal{N}$  ← Ajout d'un bruit gaussien pour exploration + clip pour rester dans des valeurs admissibles
  - 5:   Execute  $a$  in the environment
  - 6:   Observe next state  $s'$ , reward  $r$ , and done signal  $d$  to indicate whether  $s'$  is terminal
  - 7:   Store  $(s, a, r, s', d)$  in replay buffer  $\mathcal{D}$
  - 8:   If  $s'$  is terminal, reset environment state.
  - 9:   **if** it's time to update **then**
  - 10:     **for** however many updates **do**
  - 11:       Randomly sample a batch of transitions,  $B = \{(s, a, r, s', d)\}$  from  $\mathcal{D}$
  - 12:       Compute targets
$$y(r, s', d) = r + \gamma(1 - d)Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s'))$$
← Utilisation de réseaux cible (à la fois pour Q et pour  $\mu$ ) pour le calcul de la cible de Q
  - 13:       Update Q-function by one step of gradient descent using
$$\nabla_{\phi} \frac{1}{|B|} \sum_{(s, a, r, s', d) \in B} (Q_{\phi}(s, a) - y(r, s', d))^2$$
  - 14:       Update policy by one step of gradient ascent using
$$\nabla_{\theta} \frac{1}{|B|} \sum_{s \in B} Q_{\phi}(s, \mu_{\theta}(s))$$
← Gradient similaire à DPG, selon les transitions du batch
  - 15:       Update target networks with
$$\begin{aligned} \phi_{\text{targ}} &\leftarrow \rho \phi_{\text{targ}} + (1 - \rho) \phi \\ \theta_{\text{targ}} &\leftarrow \rho \theta_{\text{targ}} + (1 - \rho) \theta \end{aligned}$$
← Mise à jour "soft" des paramètres des réseaux cible
  - 16:     **end for**
  - 17:   **end if**
  - 18: **until** convergence
-

Appliquer l'algorithme aux 3 problèmes suivants:

- MountainCarContinuous-v0
- LunarLanderContinuous-v2
- Pendulum-v0

### Bonus: Q-Prop

Comparez les résultats avec l'algorithme Q-Prop (version conservative):

---

#### Algorithm 1 Adaptive Q-Prop

---

```

1: Initialize  $w$  for critic  $Q_w$ ,  $\theta$  for stochastic policy  $\pi_\theta$ , and replay buffer  $\mathcal{R} \leftarrow \emptyset$ .
2: repeat
3:   for  $e = 1, \dots, E$  do                                     ▷ Collect  $E$  episodes of on-policy experience using  $\pi_\theta$ 
4:      $s_{0,e} \sim p(s_0)$ 
5:     for  $t = 0, \dots, T-1$  do
6:        $a_{t,e} \sim \pi_\theta(\cdot | s_{t,e}), s_{t+1,e} \sim p(\cdot | s_{t,e}, a_{t,e}), r_{t,e} = r(s_{t,e}, a_{t,e})$ 
7:     Add batch data  $\mathcal{B} = \{s_{0:T-1,e}, a_{0:T-1,e}, r_{0:T-1,e}\}$  to replay buffer  $\mathcal{R}$ 
8:     Take  $E \cdot T$  gradient steps on  $Q_w$  using  $\mathcal{R}$  and  $\pi_\theta$       ← Off-Policy Critic
9:     Fit  $V_\phi(s_t)$  using  $\mathcal{B}$ 
10:    Compute  $\hat{A}_{t,e}$  using GAE( $\lambda$ ) and  $\bar{A}_{t,e} = \nabla_a Q_w(s_t, a)|_{a=\mu_\theta(s_t)}(a_t - \mu_\theta(s_t))$ .
11:    Set  $\eta_{t,e}$ 
12:    Compute and center the learning signals  $l_{t,e} = \hat{A}_{t,e} - \eta_{t,e} \bar{A}_{t,e}$       ← On-Policy Actor
13:    Compute  $\nabla_\theta J(\theta) \approx \frac{1}{ET} \sum_e \sum_t \nabla_\theta \log \pi_\theta(a_{t,e} | s_{t,e}) l_{t,e} + \eta_{t,e} \nabla_a Q_w(s_{t,e}, a)|_{a=\mu_\theta(s_{t,e})} \nabla_\theta \mu_\theta(s_{t,e})$ 
14:    Take a gradient step on  $\pi_\theta$  using  $\nabla_\theta J(\theta)$ , optionally with a trust-region constraint using  $\mathcal{B}$ 
15: until  $\pi_\theta$  converges.

```

---