Twin-Delayed Deep Deterministic Policy Gradient (TD3)

> Fujimoto, S., Van Hoof, H., & Meger, D. (2018). Addressing Function Approximation Error in **Actor-**Critic Methods. 35th International Conference on Machine Learning, ICML 2018, 4, 2587–2601.

1. Motivation

- In DDPG, we use a Critic network (q-value) to evaluate and train the Actor network (policy), hence the actor's performance strongly depends on how well Critic network estimates the action-value `q`.
- Due to boostrap learning mechanism, i.e learning by temporal-difference, the `Critic` is subject to estimation error, namely the overestimation bias and high variance build-up.
- In Double Q-Learning, we know that using an extra Q-network can help reduce this bias. This is the key idea of TD3: We use an extra network Critic, named twin-network, to perform another estimation of `q-value`. By having two Critic networks compete to each other, we can eliviate the over-confident estimation error.

2. Twin-Delayed DDPG (TD3)

- Compare two DDPG, the main change is that TD3 uses two Critic networks, namely $Q_{\theta_1}(s,a)$ and $Q_{\theta_2}(s,a)$, which are identical (twin) but with different iniatialized weights θ_1 and θ_2.
    - Let \$Q_{\bar{\theta}*1}(s',a')andQ*{\bar{\theta}_2}(s',a')\$ be their corresponding **target networks**, respectively.
    - Also, let $\pi_\phi(s)$ be the Actor network, and $\pi_{\bar{\phi}}(s')$ be the Actor-target network.

- We update the `q_target` as follows:
    - From the Replayed-Buffer, we sample a mini-batch (s,a,r,s').
    - We predict the next action for the next state s' using the Actor-target network, disturbed by a small Gaussian noise:

$$a' = \pi_{\bar{\phi}}(s') + N(0,\sigma)$$

    - We compute two action-values using the twin-Critic Target networks: q_1 = Q_{\bar{\theta}*1}(s',a'), q_2 = Q*{\bar{\theta}_2}(s',a')
    - We compute the `q_target` by

$$q_{target} = r + \gamma \min(q_1, q_2)$$

    The term $\min(q_1,q_2)$ is often referred as `Clipped Double-Q` trick.

- Then, we update the Critic Networks by:
    - Evaluate the current action-values using the twin-Critic networks:

$$q_1 = Q_{\theta_1}(s',a'), q_2 = Q_{\theta_2}(s',a')$$

    - Compute the loss to update the Critic-Networks using MSE loss:

$$Loss_Q = \frac{1}{2}[(q_1 - q_{target})^2 + (q_2 - q_{target})^2]$$

- The Actor network $\pi_\phi(s)$ is trained as done in DDPG, using either one of the Critic, e.g. Q_1:
  - We predict the action for the curernt state s using the Actor network:

$$a = \pi_\phi(s)$$

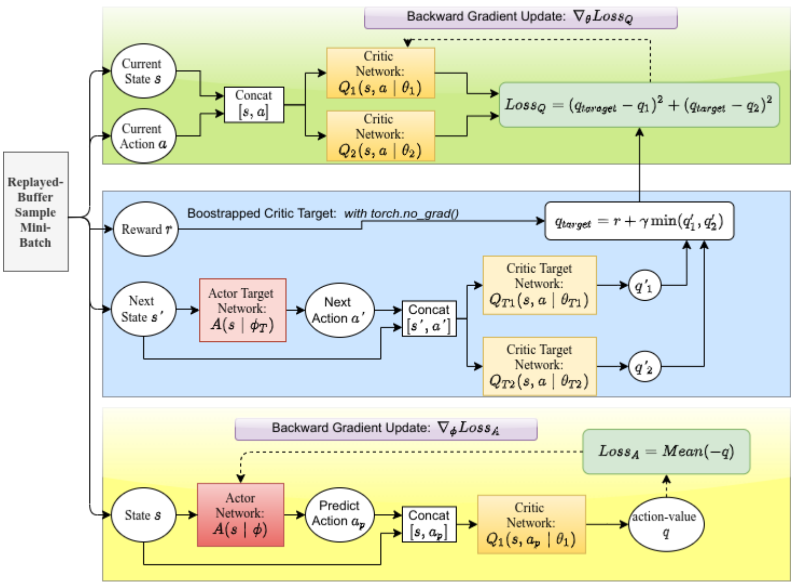
  - Loss for Actor network:

$$Loss_A = Mean(-Q_{\theta_1}(s, a))$$

3. What makes TD3 work?

- As mentioned above, using the Twin-Critic Network helps reducing the over-confident estimation for `q_target` value.
- An important trick is that, we `delay` updating the Actor and the target networks. For every two iterations, we update them **once**.

4. TD3 Pseudo Code



Algorithm 1 TD3

Initialize critic networks $Q_{\theta_1}, Q_{\theta_2}$, and actor network π_ϕ with random parameters θ_1, θ_2, ϕ
Initialize target networks $\theta_1' \leftarrow \theta_1, \theta_2' \leftarrow \theta_2, \phi' \leftarrow \phi$
Initialize replay buffer $\mathcal{B}$
for $t = 1$ **to** T **do**
  Select action with exploration noise $a \sim \pi_\phi(s) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$ and observe reward r and new state s'
  Store transition tuple (s, a, r, s') in $\mathcal{B}$

  Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$
  $\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
  $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta_i'}(s', \tilde{a})$
  Update critics $\theta_i \leftarrow \text{argmin}_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2$
  if t mod d **then**
    Update ϕ by the deterministic policy gradient:
    $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
    Update target networks:
    $\theta_i' \leftarrow \tau\theta_i + (1 - \tau)\theta_i'$
    $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$
  end if
end for

To train DDPG agent for Mountain Car Continous problem, do:

```
python tools/train.py configs/TD3/td3_mountaincar_continuous.py
```

Result after training 260 episodes: