Date	Organization	Department	Reviewed By	Approved By	Prepared By
18 August 2021	Opus Technology Limited	Artificial Intelligence			M. Mashrukh Zayed

Deep Deterministic Policy Gradient (DDPG)

Deep Deterministic Policy Gradient (DDPG) is a model-free off-policy algorithm for learning continuous actions.

- It is a **reinforcement learning technique** that combines ideas from **DPG (Deterministic Policy Gradient)** and **DQN (Deep Q-Network)**.

DDPG = DPG + DQN

- From **DQN**, it uses:

Experience Replay and Slow-learning target networks

- From **DPG**, is incorporates:

Operating over continuous action spaces

DDPG being an Actor-Critic technique:

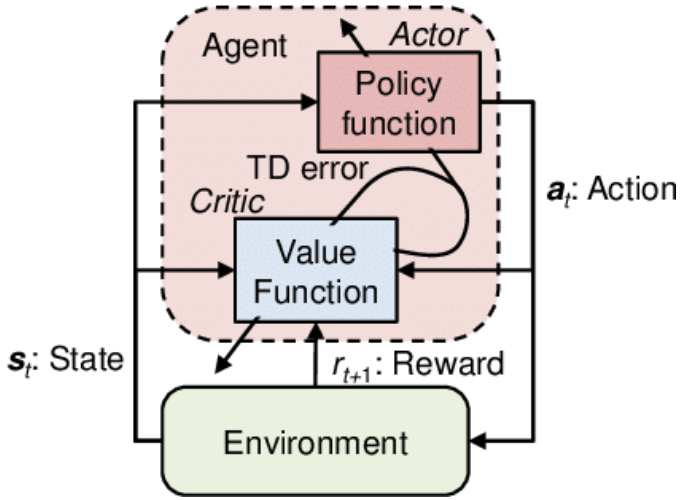
- DDPG being an **actor-critic technique** consists of two models:

  Actor: Actor is the **policy network** that takes the state as input and outputs the exact action (continuous), instead of a probability distribution over actions. The learning of the actor is based on **policy gradient approach**.

  Critic: The critic is a **Q-value network** that takes in state and action as input and outputs the Q-value. Critics **evaluate the action** produced by the actor by computing the Q-value function.

- The actor model decided **which action should be taken** and critic inform the actor **how good was the action** and how it should adjust.

- In DDPG, during the **update process of the actor**, **TD (Temporal Difference) error** from a critic is used. The critic network gets updated based on the **TD error** similar to Q-learning update rule.



DDPG being an “off”-policy method:

- Off-Policy learning algorithms **evaluate and improve a policy that is different from Policy that is used for action selection**.
  In short:

Target Policy != Behavior Policy

- DDPG is an off-policy algorithm because it uses **two separate policies** for the **exploration and updates**. It uses a **stochastic behaviour policy for the exploration** and **deterministic policy for the target update**.

- DQN works in a discrete action space (exploration) and DPG extends it to the continuous action space (target) while learning a deterministic policy.

DDPG Theories (Mathematically)

DPG Theory:

- Traditionally, **policy gradient algorithms** are being used with **stochastic policy function π(.|s)**. That means policy function π(.|s) is represented as a distribution over actions. For a given state, there will be **probability distribution** for each action in the action space. The grad objective function of the stochastic Policy gradient algorithm can be written as below:

$$\nabla_\theta J(\theta) = E[\nabla_\theta log_{\pi_\theta}(s,a) Q^w(s,a)]$$

- In **DPG**, instead of the stochastic policy, π, **deterministic policy μ(.|s)** is followed. For a given state, there will be **deterministic decision: a=μ(s)** instead of distribution over actions. We can rewrite the above equation with the deterministic policy as shown below:

$$\nabla_\theta J(\theta) = E[\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a)]$$

DQN Theory:

- Learning is a value-based off-policy **temporal difference (TD)** reinforcement learning. **Off-policy** means an agent follows a behaviour policy for choosing the action to reach the next state **s_t+1** from state **s_t**. From **s_t+1**, it uses a **policy π** that is different from behaviour policy.

- In Q-learning, we take **absolute greedy action as policy π** from the next state s_t+1.

$$Q(s,a) = r(s,a) + \gamma \max_a Q(s',a)$$

- It is important to mention the **update rule** in Q-learning. New Q value is the sum of **old Q value and TD error**.

$$Q(s_t, a_t) \leftarrow old\ Q\ value + TD\ error$$

- **TD error** is computed by **subtracting the new Q value from the old Q value**.

$$Q(s_t, a_t) \leftarrow old\ Q\ value + (new\ Q\ value - old\ Q\ value)$$
$$new\ Q\ value = [r_t + \gamma max Q(s_{t+1}, a_t)]$$

- The below equation shows the elaborate view of the updating rule.

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Big(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{new value (temporal difference target)}} - \underbrace{Q(s_t, a_t)}_{\text{old value}}\Big)}^{\text{TD error}}$$

- Overall, the off-policy learning algorithm is as follows:

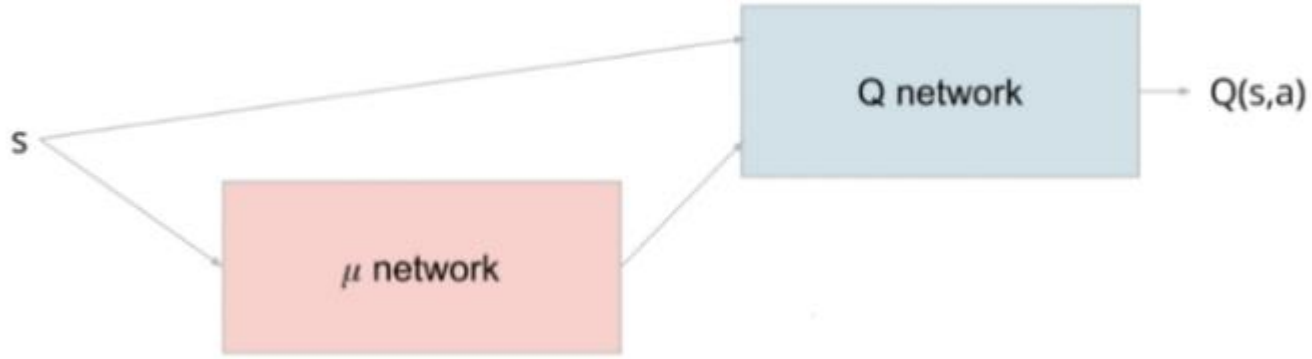
```
Initialize Q(s, a) arbitrarily
Repeat (for each episode):
   Initialize s
   Repeat (for each step of episode):
      Choose a from s using policy derived from Q
         (e.g., ε-greedy)
      Take action a, observe r, s'
      Q(s, a) <-- Q(s, a) + α [r + γ max_α Q(s', a') - Q(s, a)]
      s <-- s';
   until s is terminal
```

Network Models in DDPG

Actor-Critic Network (μ & Q):

- In DDPG, **Actor is the policy network (mu).** It is deterministic since it **directly outputs the action** (action can be continuous). In order to **promote exploration** some **Gaussian noise** is added to the action determined by the policy.

- To calculate the **Q-value** of a state, **the actor output is fed into the Q-network (Q)**.



- To **stabilize learning** we create **target networks** for both critic and actor. These target networks will have **"soft"-updates** based on main networks. This part is described next:

Target Networks:

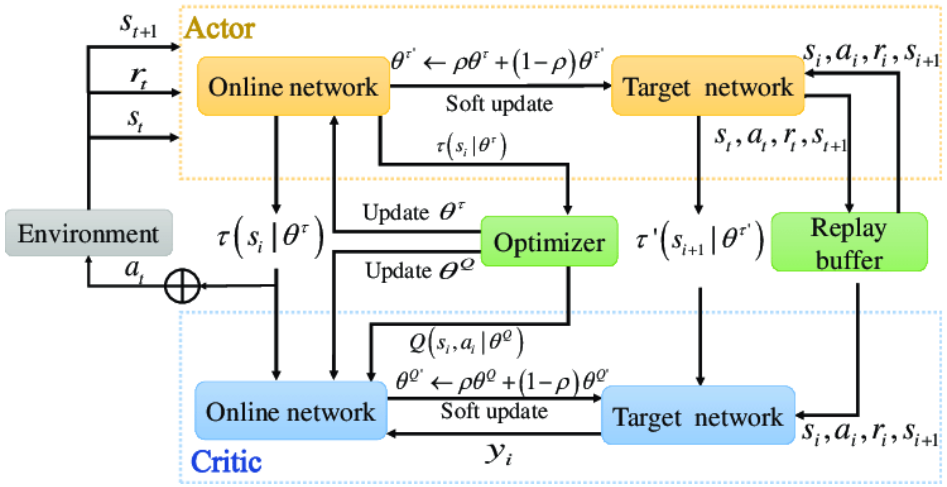
- To increase stability during training we include **target critic and actor networks** to calculate Q-value for next state in TD-error computations.

- The target networks are **delayed networks** compared to **main/current networks**. The **weights of targets** are **updated periodically** based on the main networks.

- In DQN the target gets the main network weights copied over periodically, this is known as a "**hard update"**. However, in DDPG we perform a **"soft update"** where only a fraction of main weights are transferred in the following manner:

$$\theta^{\mu}{}_{targ} \leftarrow \tau\theta^{\mu}{}_{targ} + (1-\tau)\theta^{\mu}$$
$$\theta^{Q}{}_{targ} \leftarrow \tau\theta^{Q}{}_{targ} + (1-\tau)\theta^{Q}$$

- **Tau** is a parameter that is typically chosen to be close to 1 (for example: 0.999).



Replay Buffer:

- As with other deep reinforcement learning techniques, DDPG relies on the use of Replay Buffer for stability. The replay buffer needs to maintain a balance of old and new experiences.

- **Definition:**
  The replay buffer contains a collection of experience tuples (S, A, R, S'). The tuples are gradually added to the buffer as we ar e interacting with the Environment. The simplest implementation is a buffer of fixed size, with new data added to the end of the buffer so that it pushes the oldest experience out of it.

- **Purpose:**
  A buffer of past experiences is used to stabilize training by decorrelating the training examples in each batch used to update t he neural network. This buffer records past states, the actions taken at those states, the reward received and the next state t hat was observed.

DDPG Algorithm

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

Implementation of DDPG

Content: https://colab.research.google.com/drive/1sGbtFRo8SerRtd-EP2-Hks7BDpqoCEOf?usp=sharing

---------- End ----------