Continuous control project using Unity Reacher Environment

Introduction

This report provides a brief introduction to project name 'Continuous control' offered as part of Udacity Nano degree on deep reinforcement learning. I trained an agent, which is a double-jointed arm, to maintain its position at target location for as many timesteps as possible. In the Reacher environment, the target location is a moving ball rotating around the agent. The **action size** is four with continuous nature, where a single action can take any value between *-1 to 1*. The **state space** consists of *33* variables corresponding to position, rotation, and angular velocities of the arm. There is *+1* **reward** of each step the agent's hand is at target location. For the first version of Reacher environment with one agent, the environment is considered solved when an agent accumulates an average reward of *+30* for *100* consecutive episodes.

I employed model-free algorithm called deep deterministic policy gradient (DDPG) that has shown to work well for the continuous action spaces. In the following we provide a brief overview of DDPG algorithm, the agents training performance and few pointers for future work.

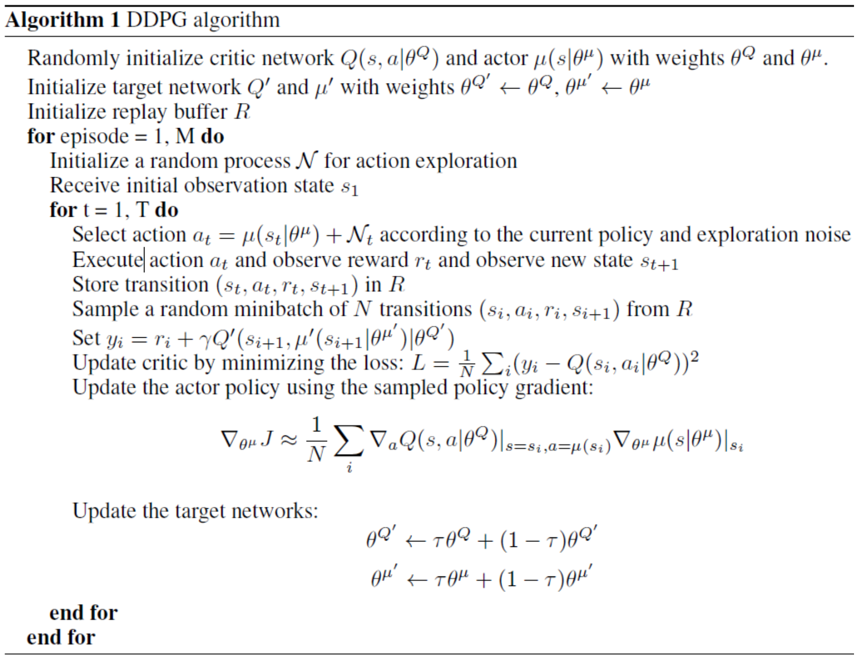
Deep Deterministic Policy Gradient (DDPG)

Deep Q network (DQN) has shown remarkable performance to solve complex tasks from high-dimensional sensory input. However, DQN can only handle discrete and low-dimensional action spaces. Deep Deterministic Policy Gradient (DDPG) proposed by [1], seeks support continuous action spaces while utilizing the actor-critic approach and insights from DQN namely replay buffer and target networks.

The DDPG work in following manner:

1. There are two neural networks in DDPG. One is called Actor and the other is Critic.
2. The actor is used to approximate the optimal policy deterministically. In other words, the actor network takes states as input and output the best believed action for given state.
3. The critic learns to evaluate the action-value function by using the best believed action.
4. Similar to DQN, the concept of experience replay and target network is utilized to stabilize the learning and to utilize knowledge from past trajectories for network training.
5. In DDPG, the target network is updated more frequently referred as soft-updates.

Following is the DDPG algorithm:



Algorithm 1 DDPG algorithm

```
Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$.
Initialize target network $Q'$ and $\mu'$ 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
```

Figure 1: DDPG Training Algorithm [1]

Agent in Action:

For the single agent of Reacher environment, the agent is trained using DDPG algorithm. I tuned several hyper-parameters namely batch size, weight decay, fully connected units, and sigma value in OUNoise. The agent was able to solve the environment 500 episodes. The environment is considered solved, when the agent is able to get average of 30+ score for 100 consecutive episodes. Figure 2 provide the training progress plot of an agent.

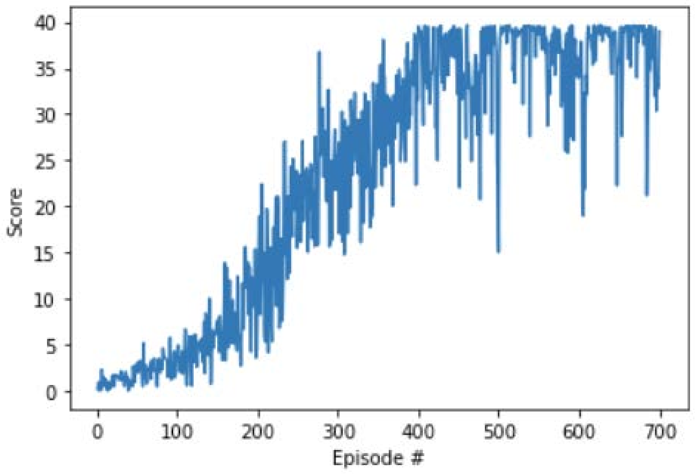


Figure 2: Average rewards using DDPG during Agent's Training

Other experiments

I also experimented with the multi-agent setting of Reacher environment, however, the agents took extremely long to train and the performance is shown to degrade after 500 episodes with 15 average scores.

Future work

The future work aims to solve the multi-agent setting of the Reacher environment. Additionally, I would also like to compare the performance of other policy-based methods such as PPO and A2C to solve the given environment.

References

[1]. Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., ... & Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.