ECE 276 Assignment 4 : Deep Deterministic Policy Gradients

November 8, 2019

The goal of this assignment is to implement deterministic policy gradient methods for an environment with continuous action spaces. For question 1 and 2 use the 2 link arm as in assignment 3. Question 2 is a bonus question.

Question 1 - DDPG

In class we went through the different tricks that was used to stabilize learning for DDPG, review the algorithm once before attempting this assignment. The actual paper for reference - Continuous control with deep reinforcement learning. For this assignment we will specifically implement the following:

1. Experience replay buffer - A buffer that stores previous state-transitions and rewards functions, which is used to train the $Q(s, a)$ and $\pi(a)$. The buffer is a set $\mathcal{D}$ which contains the previous state transitions, specifically $(s_t, a_t, r_t, s_{t+1}, \text{done})$, where s_t,a_t,s_{t+1}, r_t and done are the current state, current action, next state , reward and terminal state flag respectively. The terminal state flag is True if the transition results in termination of the environment otherwise it is false. For training initialize a replay buffer of size 10000. Before training, fill this buffer with 1000 samples with state-transitions taken from random actions.

2. Actor and Critic Networks - The actor and critic are implemented as neural networks. Use the same network parameters as mentioned in the paper.

3. Adding noise to actions : In the paper, the authors train their model by adding time correlated Ornstein-Uhlenbeck(OU) noise but more recent results sugest that uncorrelated, mean-zero Gaussian noise work perfectly fine. Thus instead of OU noise add noise from a normal distribution with 0.1 variance.

Implement the DDPG algorithm and train it on 3 different seed values for 200000 steps. Compare the amount of data used to train this model with assignment 3. Plot the return after each policy update. Evaluate this return by executing the policy on the environment without any exploration noise(Do not add this experience to the buffer). We have written a skeleton code to help you with writing the algorithm (It's optional to use).
Another useful resource about DDPG can be found on OpenAI-SpinningUp

Question 2 - TD3(Bonus)

DDPG is often susceptible to overestimated Q-values which makes it often brittle to hyperparameters. A recent paper that resolves this issue is Twin Delayed Deep Deterministic Policy Gradient (TD3). By making a few changes to the DDPG method, you can implement TD3. Train your policy using TD3 algorithm by modifying your DDPG implementation for a single seed over 200000 steps. Plot the return after each policy update. Compare your results with question 1.
OpenAI-SpinningUp has a succient write up on TD3.