[CSCI-GA 3033-090]  
Deep Decision Making and Reinforcement Learning

Assignment 3: Off-Policy Reinforcement Learning

## Instructions

This homework is designed to follow up on the lecture about Deep Q-learning. For this assignment, you will need to know about the basics of the deep Q learning algorithm we talked about in class. If you have not already, we propose you brush up on the lecture notes. We will be focusing on off-policy reinforcement learning for this assignment. The algorithms used in this homework have not been covered in the class in detail so you will have to learn from them using research papers or blogs on the internet.

**You are ALLOWED to discuss this homework with your classmates at the level of general solution strategies or tips. However, any work that you submit must be entirely your own, which includes your own unique code and your own unique written answers.**

**You are NOT ALLOWED to use any large language models (LLMs) or other AI-assisted tools.**

If you find errors in the homework assignment or have public questions, please post in the Campuswire “#questions-hw” channel or ask during instructor/TA office hours.

## Points

| Q1 | RL training only |  | 10 pts |
| --- | --- | --- | --- |
| Q2 | BC training and RL finetuning |  | 10 pts |
| Q3 | RL variants |  | 5 pts |
| Bonus | SAC implementation |  | 5 pts |
|  |  |  | **25 pts total**  **30 pts max (with bonus)** |

## Coding Questions

In class, we learned about the Deep Q-Network (DQN) Learning algorithm, which is considered the first large scale success for any deep reinforcement algorithm. This method is quite dated now, but for a lot of algorithms used today, the roots can be traced back to DQN.

In this assignment, we will be focusing on an actor-critic based off-policy RL algorithm called [Deep Deterministic Policy Gradients (DDPG)](https://arxiv.org/abs/1509.02971). In DDPG, the authors use [Ornstein-Uhlenbeck Process](https://en.wikipedia.org/wiki/Ornstein%E2%80%93Uhlenbeck_process) to add noise to the action output. However, for simplicity, we have used a constant standard deviation for exploration in this assignment. In addition to vanilla RL, we will also be testing a combination of behavior cloning (BC) and RL to enhance the online sample efficiency further.

**Code**:

<https://drive.google.com/file/d/1hI5vPUEjj8LVwQLBWHlrxk6Hu_qCclWe/view?usp=drive_link>

You are only required to modify the code provided in the scripts in *policy/agent* directory. Find out the TODOs and complete them. The virtual env can be set up using *conda\_env.yml* (from previous assignments) and instructions for running the code have been included in *instructions.md*.

**Note: Since *particle-envs* has been modified, please either “pip install -e” it again if that is how you installed it, or replace the directory if you merely moved it to be on the python search path.**

**Environment:**

The environment conforms to the OpenAI Gym API, which you can learn more about at <https://github.com/openai/gym#api>. It is a goal-reaching environment where the agent is spawned at a start location and is tasked with reaching a goal location.

This is the same environment as Assignment 1 with minor changes in the episode length (changed to 50) and reward functions.

**Deliverables:**

* Within PDF write-up, any written answers and plots.
* Within compressed ZIP folder, the code files you changed and any code files for the bonus question. Please name your submission ZIP folder as *<net\_ID>\_assignment3.zip*

### 1. RL training only

Complete the TODOs in the following scripts in the *policy/agent* directory - *networks/actor.py*, *networks/critic.py*, and *networks/rl.py*. Provide the training and evaluation curves in a PDF and report your observations.

### 2. BC training and RL finetuning

One way to enhance the sample efficiency of RL is to pre-train the policy with behavior cloning and fine-tune online with RL. The actor loss used in this case is a combination of the Q-learning loss and a BC loss.

Complete the TODO sections in *policy/agent/bcrl.py*. The data for BC has been provided in *bc.pkl*. Provide the training and evaluation curves in a PDF and compare the performances of RL and BCRL. Provide an intuition about any differences in performance that you may observe.

### 3. RL variants

[Double Q-learning](https://paperswithcode.com/method/double-q-learning) and [Dueling DQN](https://arxiv.org/abs/1511.06581) are a couple of techniques that have been used to improve the performance of DQN. Briefly describe how each of the two methods are different from the provided code. Do you think these methods would also help improve performance in the provided actor-critic based off-policy RL algorithm? Explain.

## Bonus

In this assignment, we have used a fixed standard deviation in the actor for exploration. [Soft-actor critic (SAC)](https://arxiv.org/abs/1812.05905) is an off-policy algorithm that learns the standard deviation over the course of online learning. Implement SAC using the current codebase.  
(**Hint:** You will need to change the actor updater and add a temperature variable to the RL agent)