ECE433/COS435 Introduction to RL
Assignment 7: MaxEnt
Spring 2024

Fill me in

Your name here.

Due April 9, 2024

Collaborators

Fill me in

Please fill in the names and NetIDs of your collaborators in this section.

Instructions

Writeups should be typesetted in Latex and submitted as PDFs. You can work with whatever tool you like for the code, but **please submit the asked-for snippet and answer in the solutions box as part of your writeup. We will only be grading your write-up.** Make sure still also to attach your notebook/code with your submission.

We recommend you collaborate in pairs if you find the workload of this assignment overwhelming.

Question 1. Implementing TD3 with MaxEnt

Using the solution for TD3 in Homework 5 (we'll give it to you), we are going to re-write it using the MaxEnt framework. Your policy will now be a **Gaussian** distribution that gets sampled instead of a deterministic continuous variable. See `https://pytorch.org/docs/stable/distributions.html` for more information about PyTorch distributions. We will also provide a set of hints below that may be useful for you.

1. Be careful to distinguish between torch.distributions.distribution.Distribution.rsample() and torch.distributions.distribution.Distribution.sample().

2. The careful reader may notice that the Gaussian distribution is unbounded, but your environment (MountainCar) has a bounded action space. Be sure to clip your outputs using an activation function or `torch.clip`.

3. You **do not** need to factor in this "clipping" transformation in your entropy computation. Technically, you need to perform a change of variable to compute the correct entropy, but your solution should still work by only using the entropy of the Gaussian as the entropy in your expressions.

4. Generally, you are supposed to leave the mean and variance μ, σ^2 unbounded. However, you may notice that our action space is bounded here, so you might find it easier to just bound these terms with an activation function to avoid exploding terms.

5. You may want to scale your entropy with a constant α as a tunable hyperparameter.

Question 1.a

Rewrite the `Actor_TD3` class such that the Actor now outputs a **Gaussian distribution** in its forward pass. You may find it easier to handle sampling actions and computing the entropy/log probabilities **outside of the forward()** function. Paste your class below:

Solution

```
class Actor_TD3:
    pass
```

Question 1.b

Rewrite the `select_action()` function to sample from your policy distribution instead of outputting a deterministic action.

Solution

```
def select_action():
    pass
```

Question 1.c

Re-write your actor and critic loss to support your MaxEnt implementation. Paste the entire code of `train()` below.
Hint: Look at your target Q and actor loss computations for DDPG and see what to change based on the MaxEnt and Bellman equations.

Solution

Your solution here...

```python
def train():
    pass
```

Question 1.d

Insert the reward v.s. episode curve below. If you notice that your model is unstable but sometimes has positive reward, try tuning some parameters. Your method should be able to converge to positive reward.

Solution

Your figure here...