Assignment 1: Imitation Learning

Due September 30, 11:59 pm

The goal of this assignment is to experiment with imitation learning, including direct behavior cloning and the DAgger algorithm. In lieu of a human demonstrator, demonstrations will be provided via an expert policy that we have trained for you. Your goals will be to set up behavior cloning and DAgger, and compare their performance on a few different continuous control tasks from the OpenAI Gym benchmark suite. Turn in your report and code as described in Section 4.

The starter-code for this assignment can be found at the following Rivanna path:

/project/SDS/instructional/ds6559_fl23/hw_ds6559_fl2023/hw1

You have the option of running the code either on Google Colab or on your own machine. Please refer to the README for more information on setup.

1 Analysis

Consider the problem of imitation learning within a discrete MDP with horizon T and an expert policy π^*. We gather expert demonstrations from π^* and fit an imitation policy π_θ to these trajectories so that

$$\mathbb{E}_{p_{\pi^*}(s)}\pi_\theta(a \neq \pi^*(s) \mid s) = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{p_{\pi^*}(s_t)}\pi_\theta(a \neq \pi^*(s_t) \mid s_t) \leq \epsilon,$$

i.e., the expected likelihood that the learned policy π_θ disagrees with the expert π^* within the training distribution p_{π^*} of states drawn from random expert trajectories is at most ϵ.

For convenience, the notation $p_\pi(s_t)$ indicates the state distribution under π at time step t while $p(s)$ indicates the state marginal of π across time steps, unless indicated otherwise.

1. Show that $\sum_{s_t}\left|p_{\pi_\theta}(s_t) - p_{\pi^*}(s_t)\right| \leq 2T\epsilon$
2. Consider the expected return of the learned policy π_θ for a state-dependent reward $r(s_t)$, where we assume the reward is bounded with $|r(s_t)| \leq R_{\max}$:

$$J(\pi) = \sum_{t=1}^{T} \mathbb{E}_{p_\pi(s_t)} r(s_t)$$

   (a) Show that $J(\pi^*) - J(\pi_\theta) = \mathcal{O}(T\epsilon)$ when the reward only depends on the last state, i.e., $r(s_t) = 0$ for all $t < T$.

   (b) Show that $J(\pi^*) - J(\pi_\theta) = \mathcal{O}(T^2\epsilon)$ for an arbitrary reward.

2 Editing Code

The starter code provides an expert policy for each of the MuJoCo tasks in OpenAI Gym. Fill in the blanks in the code marked with Todo to implement behavioral cloning. A command for running behavioral cloning is given in the Readme file. We recommend that you read the files in the following order. For some files, you will need to fill in blanks, labeled TODO.

- scripts/run hw1.py
- infrastructure/rl trainer.py
- agents/bc agent.py (another read-only file)
- policies/MLP policy.py
- infrastructure/replay buffer.py
- infrastructure/utils.py
- infrastructure/pytorch utils.py

3 Behavioral Cloning

1. Run behavioral cloning (BC) and report results on two tasks: the Ant environment, where a behavioral cloning agent should achieve at least 30% of the performance of the expert, and one environment of yourchoosing where it does not. Here is how you can run the Ant task:

```
python ds6559/scripts/run_hw1.py \
  --expert_policy_file ds6559/policies/experts/Ant.pkl \
  --env_name Ant-v4 --exp_name bc_ant --n_iter 1 \
  --expert_data ds6559/expert_data/expert_data_Ant-v4.pkl \
 --video_log_freq -1
```

When providing results, report the mean and standard deviation of your policy's return over multiple rollouts in a table, and state which task was used. When comparing one that is working versus one that is not working, be sure to set up a fair comparison in terms of network size, amount of data, and number of training iterations. Provide these details (and any others you feel are appropriate) in the table caption.

Note: What "report the mean and standard deviation" means is that your eval batch size should be greater than ep_len, such that you're collecting multiple rollouts when evaluating the performance of your trained policy. For example, if ep_len is 1000 and eval_batch size is 5000, then you'll be collecting approximately 5 trajectories (maybe more if any of them terminate early), and the logged Eval_AverageReturn and Eval_StdReturn represents the mean/std of your policy over these 5 rollouts. Make sure you include these parameters in the table caption as well.

Tip: To generate videos of the policy, remove the flag --video_log _freq -1 However, this is slower, and so you probably want to keep this flag on while debugging.

2. Experiment with one set of hyperparameters that affects the performance of the behavioral cloning agent, such as the amount of training steps, the amount of expert data provided, or something that you come up with yourself. For one of the tasks used in the previous question, show a graph of how the BC agent's performance varies with the value of this hyperparameter. In the caption for the graph, state the hyperparameter and a brief rationale for why you chose it.

4 DAgger

1. Once you've filled in all of the TODO commands, you should be able to run DAgger.

```
--python ds6559/scripts/run_hw1.py \
--expert_policy_file ds6559/policies/experts/Ant.pkl \
--env_name Ant-v2 --exp_name dagger_ant --n_iter 10 \
--do_dagger --expert_data ds6559/expert_data/expert_data_Ant-v2.pkl \
--video_log_freq -1
```

2. Run DAgger and report results on the two tasks you tested previously with behavioral cloning (i.e., Ant + another environment). Report your results in the form of a learning curve, plotting the number of DAgger iterations vs. the policy's mean return, with error bars to show the standard deviation. Include the performance of the expert policy and the behavioral cloning agent on the same plot (as horizontal lines that go across the plot). In the caption, state which task you used, and any details regarding network architecture, amount of data, etc. (as in the previous section).

5 Turning it in

Make a PDF report containing the mathematical proofs of Question 1 and the exported figures for Questions 3 and 4.