Purdue CS 558 Assignment 2: Imitation Learning

Spring 2025
Due March 12, 2025

Part 1: Behavior Cloning

The starter code for this assignment is given in the attached zip file. You have the option of running the code either on your own machine (recommended) or on a cloud-based service such as Google Colab. Please refer to the README for more information on setup.

The starter code provides an expert policy for each of the MuJoCo tasks in Gymnasium. 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.md, within the corresponding source folder. We recommend that you read the files in the following order. You will need to fill in some components, labeled `TODO` in the code, and listed out here.

- `scripts/run_hw2.py`
  - You will not have to modify this file, but it does point you to components you will modify.
- `infrastructure/rl_trainer.py`
  – Write train_agent().
  – Write collect_training_trajectories().
  – Write do_relabel_with_expert() - only needed for DAgger.
- `agents/bc_agent.py`
  - You will not have to modify this file, but it does point you to components you will modify.
- `policies/MLP_policy.py`
  – Write get_action().
  – Write forward().
  – Write update().

- `infrastructure/replay_buffer.py`
  - Write sample_random_data().
- `infrastructure/utils.py`
  – Write sample_trajectory().
  – Write sample_trajectories().
  – Write sample_n_trajectories().
- `infrastructure/pytorch_utils.py`
  – Write build_mlp().

1. Expert data analysis [1 pt]

We've provided you with expert policy data in the `expert_data/` folder of the source code. This will be the data you use to train your behavior cloning agent. For each environment, report the mean and standard deviation of return over two trajectories of the expert data.

Note: There are four environments we have collected data from: Ant-v4, Walker2d-v4, Hopper-v4 and HalfCheetah-v4.

2. Behavior cloning [2 pts]

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 your choosing where it achieves less than 30% expert performance. Here is how you can run the Ant task:

```
python cs558/scripts/run_hw1.py \
--expert_policy_file cs558/policies/experts/Ant.pkl \
--env_name Ant-v4 --exp_name bc_ant --n_iter 1 \
--expert_data cs558/expert_data/expert_data_Ant-v4.pkl \
--video_log_freq -1
```

Tip: To reach at least the 30% of the performance of the expert, you can tune these parameters: `train_batch_size` and `num_agent_train_steps_per_iter`. 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.

Tip: Training logs and videos are logged as events to Tensorboard. To view them, run tensorboard and set `logdir` to `data/`. See README.md for more details on visualization.

3. Hyperparameter sensitivity analysis [1 pt]

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 this graph, report both the mean and standard deviation of return over at least five rollouts. In the caption for the graph, state the hyperparameter and a brief rationale for why you chose it.

Part 2: DAgger

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

```
python cs558/scripts/run_hw2.py \
--expert_policy_file cs558/policies/experts/Ant.pkl \
--env_name Ant-v4 --exp_name dagger_ant --n_iter 10 \
--do_dagger --expert_data cs558/expert_data/expert_data_Ant-v4.pkl \
--video_log_freq -1
```

Note: Although in class we discussed a decaying probability to sample actions from the teacher/student (β in the algorithm from the lecture notes), here you can assume that at the first timestep $\beta = 1$ such that all states come from the teacher and in all subsequent timesteps $\beta = 0$ such that all states come from the student. When collecting corrective

actions from the teacher in `do_relabel_with_expert()` you should correct all actions in a batch, as in the algorithm discussed during the lecture.

1. DAgger and comparison to BC [2 pts]

Run DAgger and report results on the two tasks you tested previously with behavioral cloning (i.e., Ant and 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).

Submission instructions

Using Latex or another text editor, make a PDF report containing: a table of results from Question 1.1, a table of results from Question 1.2, a figure for Question 1.3. and a figure with results from question 2.1.

In order to turn in your code and experiment logs, create a folder that contains the following:

- A folder named run logs with experiments for both the behavioral cloning (part 2, not part 3) exercise and the DAgger exercise. Note that you can include multiple runs per exercise if you'd like, but you must include at least one run (of any task/environment) per exercise. These folders can be copied directly from the cs558/data folder into this new folder. **Important: Disable video logging for the runs that you submit, otherwise the files size will be too large!** You can do this by setting the flag `--video_log_freq -1`.
- The cs558 folder with all the .py files, with the same names and directory structure as the original homework repository. Also include the commands (with clear hyperparameters) that we need in order to run the code and produce the numbers that are in your figures/tables in a RUN.md file in the form of a README file.

Disclaimer: *This assignment is adapted from a homework exercise originally developed for CMU's 16-831 taught by Guanya Shi. We gratefully acknowledge the work and contributions of the original instructors and course developers. Any similarities in content or structure are intentional and provided with full credit to the original source.*