Assignment 1: Imitation Learning

Released: September 2
Due: September 18, 11:59pm

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. The entire assignment is normalized to 15 points (pt), the points for each section and question are noted. Turn in your report and code as described in Section 3.

The starter-code for this assignment can be found at

`https://github.com/LeCAR-Lab/16831-F25-HW`

You have the option of running the code either on Google Colab or on your own machine. We recommend running it on a system with linux. Please refer to the `README` for more information on setup.

1 Behavioral Cloning (9.75 pt)

1. 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.md" for HW1, within the HW1 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_hw1.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()`.
   - `agents/bc_agent.py` (read-only file) - 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_util.py`
     - Write `build_mlp()`.

2. (1.5 pt) We've provided you with expert policy data in the `expert_data/` folder in the github repository. This will be the data you use to train your behavior cloning agent implemented above. **For each environment, report the mean and standard deviation of return over two trajectories of the expert data in Table 1**.

   Note: there are five environments we have collected data from: Ant-v2, Humanoid-v2, Walker2d-v2, Hopper-v2 and HalfCheetah-v2.

3. (5.25 pt) 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% of the expert performance. Here is how you can run the Ant task (change name of expert params for other environments):

```
python rob831/scripts/run_hw1.py \
  --expert_policy_file rob831/policies/experts/Ant.pkl \
  --env_name Ant-v2 --exp_name bc_ant --n_iter 1 \
  --expert_data rob831/expert_data/expert_data_Ant-v2.pkl \
  --learning_rate 4e-3 --n_layers 5 --video_log_freq -1
```

   Tip: If your policy is not reaching at least 30% of the performance of the expert, try tuning hyperparameters

   When providing results, **report the mean and standard deviation of your policy's return over multiple rollouts in Table 2, 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.

4. (3 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 Figure 1. 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.**

2 DAgger (5.25 pt)

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

```
python rob831/scripts/run_hw1.py \
  --expert_policy_file rob831/policies/experts/Ant.pkl \
  --env_name Ant-v2 --exp_name dagger_ant --n_iter 8 \
  --do_dagger --expert_data rob831/expert_data/expert_data_Ant-v2.pkl \
  --n_layers 3 --video_log_freq -1
```

2. (5.25 pt) Run DAgger and report results on the two tasks you tested previously with behavioral cloning (i.e., Ant + another environment) using the command above. **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 in Figure 2. Include the performance of the expert policy and the behavioral cloning agent on the same plot for a single environment (as horizontal lines that go across the plot). Follow the submission template to plot the Ant-v2 environment on the left and the other environment you choose on the right. In the caption, state which task you used, and any details regarding network architecture, amount of data, etc. (as in the previous section).

3 Submission

1. **Submitting the PDF.** We have provided a Latex PDF template in the github repository for you to use. Using this template, make a PDF report containing: Table 1 for a table of results from Question 1.2, Table 2 for Question 1.3, Figure 1 graph for Question 1.4, and Figure 2 with results from Question 2.2.

   Make sure you write the Andrew IDs of any students that you collaborated with in your report. You do not need to write anything else in the report, just include the figures with captions as described in each question above.

2. **Submitting the code and experiment runs.** In order to turn in your code and experiment logs, create a folder, `submit.zip`, 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 `rob831/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 `rob831` 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 (e.g. run "python run_hw1_behavior_cloning.py –ep_len 200" to generate the numbers for Section 2 Question 2) in a `RUN.md` file in the form of a README file.

As an example, the unzipped version of your submission should result in the following file structure. **Make sure that the submit.zip file is below 15MB and that they include the prefix `q1_` and `q2_`.**

```
submit.zip
├── run_logs
│   ├── q1_bc_ant
│   │   └── events.out.tfevents.1567529456.e3a096ac8ff4
│   ├── q2_dagger_ant
│   │   └── events.out.tfevents.1567529456.e3a096ac8ff4
│   └── ...
├── rob831
│   ├── agents
│   │   ├── bc_agent.py
│   │   └── ...
│   ├── policies
│   │   └── ...
│   └── ...
├── README.md
└── ...
```

Note: If you are a Mac user, **do not use the default "Compress" option to create the zip**. It creates artifacts that the autograder does not like. You may use `zip -vr submit.zip submit -x "*.DS_Store"` from your terminal.

Turn in your assignment on **Gradescope**. Upload the zip file with your code and log files to **HW1 - Code**, and upload the PDF of your report to **HW1 - Written**.