Deep RL Assignment 1: Imitation Learning

Fall 2019

due September 16th, 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 `https://github.com/berkeleydeeprlcourse/homework_fall2019`. Follow the instructions in the Readme file to setup the codebase.

Section 1. Behavioral Cloning

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 file.

   The following files have blanks in them and can be read in this order:

   - `scripts/run_hw1_behavior_cloning.py`
   - `infrastructure/rl_trainer.py`
   - `agents/bc_agent.py`
   - `policies/MLP_policy.py`
   - `infrastructure/replay_buffer.py`
   - `infrastructure/utils.py`
   - `infrastructure/tf_utils.py`

2. Run behavioral cloning (BC) and report results on two tasks: one task where a behavioral cloning agent achieves at least 30% of the performance of the expert, and one task where it does not. When providing results, report the mean and standard deviation of the return over multiple rollouts in a table, and state which task was used. Be sure to set up a fair comparison, in terms of network size, amount of data, and number of training iterations, and provide these details (and any others you feel are appropriate) in the table caption.

   Tip: to speed up run times, the video logging can be disabled by setting `--video_log_freq -1`

3. Experiment with one set of hyperparameter that affects the performance of the behavioral cloning agent, such as the number of demonstrations, the number of training epochs, the variance of the expert policy, 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, and state the hyperparameter and a brief rationale for why you chose it in the caption for the graph.

Section 2. DAgger

1. Implement DAgger by filling out all the remaining blanks in the code marked with `Todo`. A command for running DAgger is provided in the Readme file.

2. Run DAgger and report results on one task in which DAgger can learn a better policy than behavioral cloning. 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. In the caption, state which task you used, and any details regarding network architecture, amount of data, etc. (as in the previous section).

Section 3. Turning it in.

1. **Submitting the PDF** Make a PDF report containing: Table 1 for a table of results from Question 1.2, and Figure 1 for Question 1.3. and Figure 2 with results from question 2.2.

   You do not need to write anything else in the report, just include the figures with captions as described in each question above. See the handout at `http://rail.eecs.berkeley.edu/deeprlcourse/static/misc/viz.pdf` for notes on how to generate plots.

2. **Submitting the code and experiment runs** In order to turn in your code and experiment logs, create a folder that contains the following:

   - A folder named `run_logs` with **at most one folder per environment** for either the behavioral cloning (part 2, not part 3) or DAgger exercise. These folders can be copied directly from the `cs285/data` folder. **Important: Disable video logging for the runs that you submit, otherwise the files ize will be too large! You can do this by setting the flag** `--video_log_freq -1`
   - The `cs285` folder with all the `.py` files, with the same names and directory structure as the original homework repository. Also include any special instructions we need to run it to produce each of your figures or tables (e.g. "run python myassignment.py -sec2q1" to generate the result for Section 2 Question 1) 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.**

```
submit.zip
├── run_logs
│   ├── dagger_Ant-v2_03-09-2019_16-50-56
│   │   ├── events.out.tfevents.1567529456.e3a096ac8ff4
│   │   ├── checkpoint
│   │   ├── policy_itr_0.data-00000-of-00001
│   │   └── ...
│   └── bc_Ant-v2_03-09-2019_16-50-56
│       ├── events.out.tfevents.1567529456.e3a096ac8ff4
│       ├── checkpoint
│       ├── policy_itr_0.data-00000-of-00001
│       └── ...
├── cs285
│   ├── agents
│   │   ├── bc_agent.py
│   │   └── ...
│   ├── policies
│   │   └── ...
│   └── ...
├── README.md
└── ...
```

3. 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**.