CS224R Spring 2023 Homework 1
Imitation Learning

Due 4/19/2023

SUNet ID: 06661759
Name: Paul-Emile Giacomelli
Collaborators: Yoni Gozlan & Josselin Sommerville

By turning in this assignment, I agree by the Stanford honor code and declare that all of this is my own work.

Problem 1: Behavior 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 your choosing where it does not. A policy that achieves greater than 30% of the expert on the Ant task will receive full credit on the autograder. Here is how you can run the Ant task:

```
python cs224r/scripts/run_hw1.py \
    --expert_policy_file cs224r/policies/experts/Ant.pkl \
    --env_name Ant-v4 --exp_name bc_ant --n_iter 1 \
    --expert_data cs224r/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.

Include your table here.

Environment	Eval Average Return	Eval Std Return	% of expert performance
Ant-v4	3589.07	1460.94	76.14
Walker2d-v4	407.15	367.51	7.31

Table 1: These results were obtained using Kullback–Leibler divergence as a loss function between the predicted and the expert action distributions.
The hyperparameters are:
-- train_batch_size 100, -- batch_size 1000,
-- eval_batch_size 5000, -- n_iter 1, -- ep_len 1000, -- n_layers 2, -- size 64,
-- learning_rate 5e3, -- max_replay_buffer_size 1000000

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. State the hyperparameter and a brief rationale for why you chose it.

   Include your chosen hyperparameter, plot, and rationale here.

   I chose the number of layers n_layers as the hyperparameter. I wanted to see the impact of the number of layers on both the performance of the behaviour cloning agent and on the computation time.

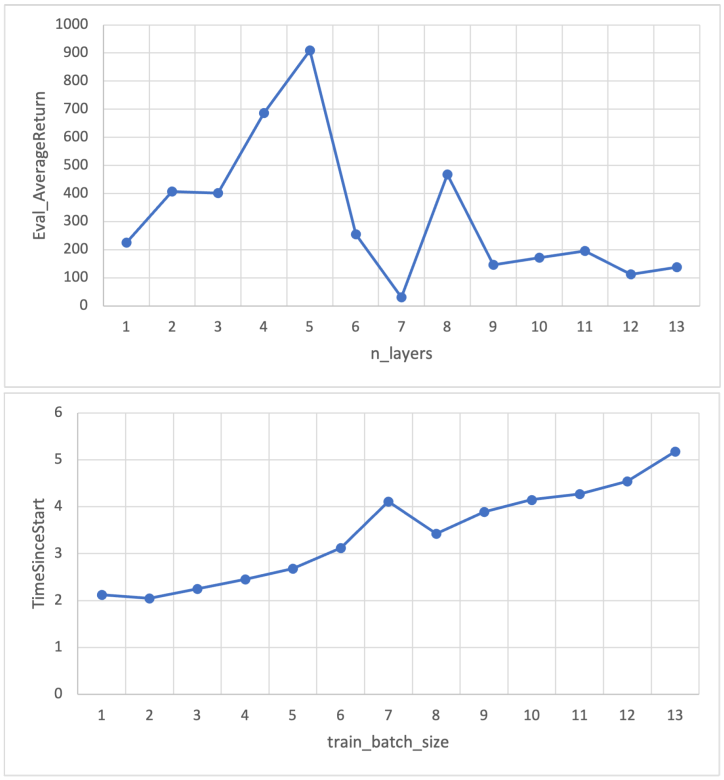

Figure 1: Evolution of Eval_AvergageReturn and TimeSinceStart with respect to n_layers on Walker2d-v4

We can see that the performance is clearly not proportional to the number of layers. However, it is *almost* the case for the computation time, which is expected as we increase the number of weights linearly as we kept the size constant.
The hyperparameters are:
-- train_batch_size 100, -- batch_size 1000,
-- eval_batch_size 5000, -- n_iter 1, -- ep_len 1000, -- size 64,
-- learning_rate 5e3, -- max_replay_buffer_size 1000000

Problem 2: DAgger

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

```
python cs224r/scripts/run_hw1.py \
  --expert_policy_file cs224r/policies/experts/Ant.pkl \
  --env_name Ant-v4 --exp_name dagger_ant --n_iter 10 \
  --do_dagger \
  --expert_data cs224r/expert_data/expert_data_Ant-v4.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).

   Include the plots of the two tasks here.

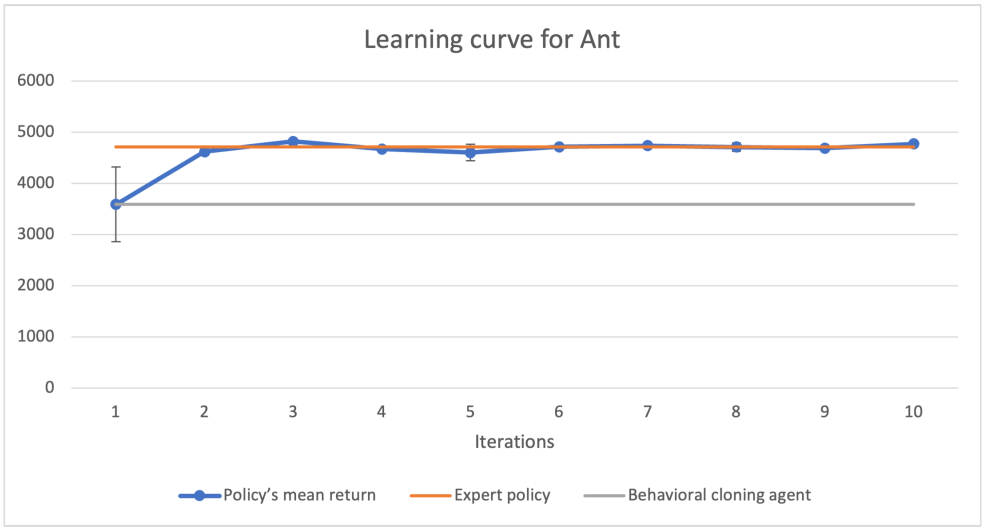

Figure 2: Evolution of eval average return with training iterations on Ant-v4.

The hyperparameters are:
-- train_batch_size 100, -- batch_size 1000,
-- eval_batch_size 5000, -- n_iter 10, -- ep_len 1000, -- n_layers 2, -- size 64,
-- learning_rate 5e3, -- max_replay_buffer_size 1000000

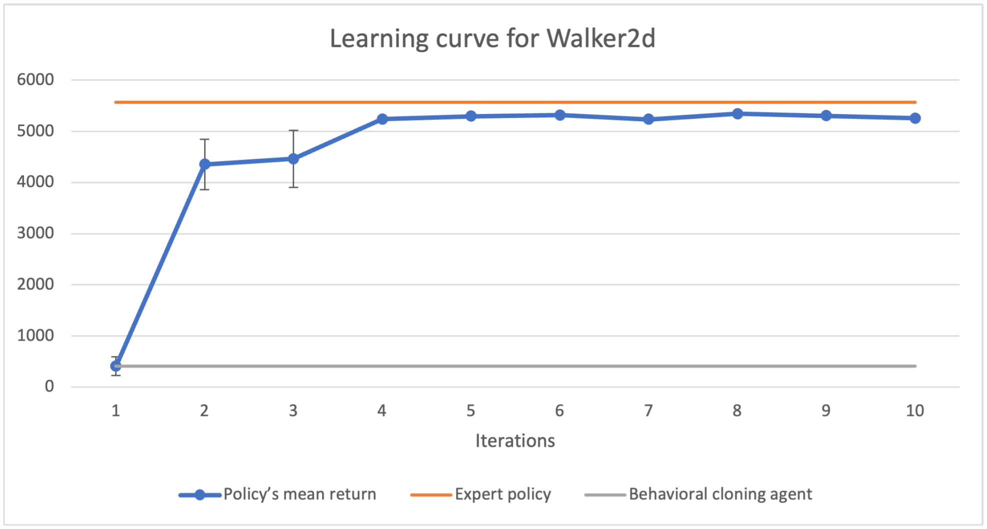

Figure 3: Evolution of eval average return with training iterations on Walker2d-v4.

The hyperparameters are the same:
-- train_batch_size 100, -- batch_size 1000,
-- eval_batch_size 5000, -- n_iter 10, -- ep_len 1000, -- n_layers 2, -- size 64,
-- learning_rate 5e3, -- max_replay_buffer_size 1000000