Homework 1

Author:
Reece D. Huff

CS 285: Deep Reinforcement Learning

Dr. Sergey Levine

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

---

1 Behavioral Cloning

Environment	Behavioral Cloning Policy		Expert Policy	
	`Eval_AverageReturn`	`Eval_StdReturn`	`Train_AverageReturn`	`Train_StdReturn`
Ant	4735.765	97.249	4713.653	12.200
Walker	340.427	321.987	5566.846	9.238

Table 1: All of the values collected for this table were using the default values of hyperparameters of behavioral cloning (see the `run_hw1.py` script to see the defaults), expect that the `ep_len` and `eval_batch_size` were set to 1000 and 5000, respectively, such that the averages and standard deviations above were across 5 rollouts.

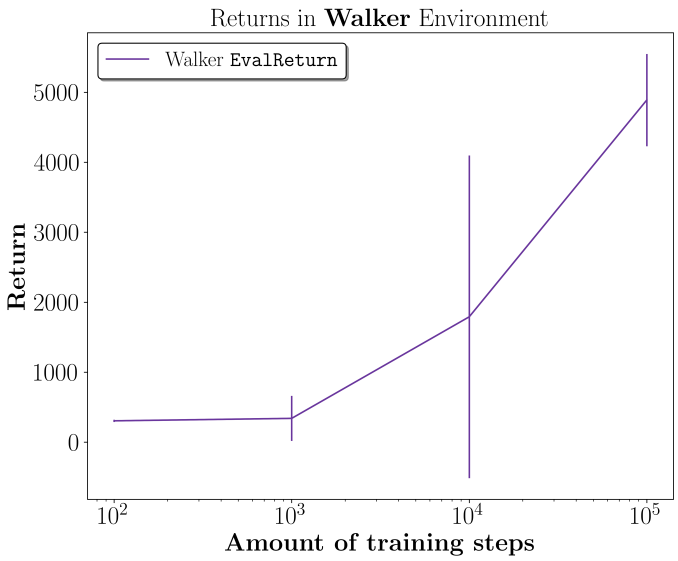
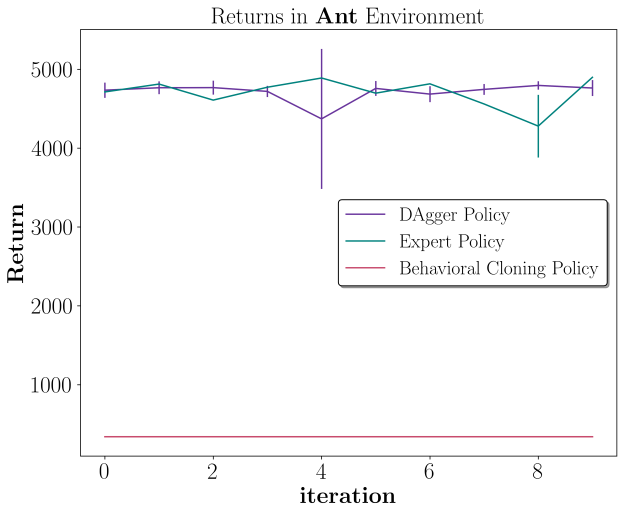


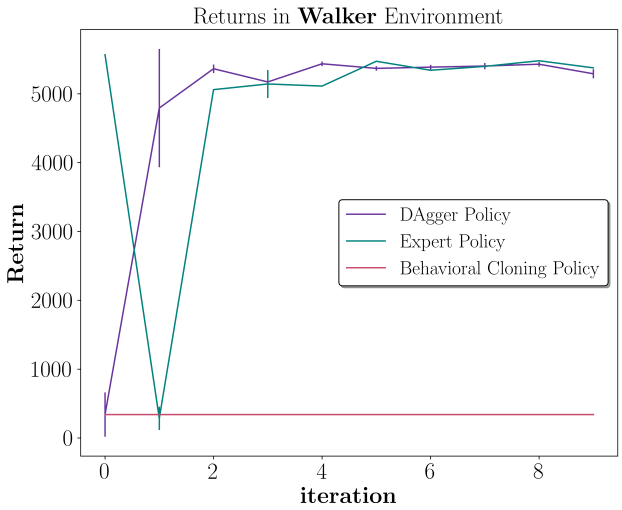
Figure 1: All of the values collected for this figure were using the default values of hyperparameters of behavioral cloning (see the `run_hw1.py` script to see the defaults), expect that the `ep_len` and `eval_batch_size` were set to 1000 and 5000, respectively, such that the averages and standard deviations above were across 5 rollouts.

The plot above is a hyperparameter tuning plot of behavioral cloning in the Walker environment. I varied the amount of training steps because I figured that the relatively low reward in the Walker environment compared to the Ant environment (see Table 1) may be explained by an insufficient training set size. We see that increasing the training set size results in an increase in reward.

2 DAgger



(a) The DAgger policy, the expert policy, and the behavioral cloning policy mean ± the standard deviation of **return** in the **Ant** environment.



(b) The DAgger policy, the expert policy, and the behavioral cloning policy mean ± the standard deviation of **return** in the **Walker** environment.

Figure 2: All of the values collected for this figure were using the default values of hyperparameters of behavioral cloning (see the `run_hw1.py` script to see the defaults), expect that the `ep_len` and `eval_batch_size` were set to 1000 and 5000, respectively, such that the averages and standard deviations above were across 5 rollouts for each iteration.
It is quite interesting to note the sudden dip in the expert policy in (b). I am not sure how to explain this phenomenon.