HW1_Write_Up

Chris Gagne SSID=25952284

September 12, 2017

0.1 Q 3.1 Behavioral Cloning (Ant and Cheetah Envs)

Environments: Cheetah-v1, Ant-v1

Training Data: I created 98 rollouts with the cheetah expert and ant, capping each episode at 100 time steps. I capped at 100 time steps because I wanted to balance keeping the training data small with having alot of examples of the cheetah getting started - this seems like the hardest part to learn. This created 9700 (s,a) pairs for training for cheetah and ant.

Network: 2 hidden layers with 64 units each (like the expert) and using tanh nonlinearity

Training: I used mean square error loss. I did full gradient descent (ie. used the full training set for each gradient step). I also experimented a bit with batch gradient descent, which is probably the way to go. I normalized training observations using mean, std for entire dataset. I normalized the observations for testing in the gym. I did 10,000 gradient steps for each. I used a learning rate = 0.2

Interleaving Gym with gradient steps: Every 10 gradient steps (still with the full dataset), I run an episode in the gym and store the results.

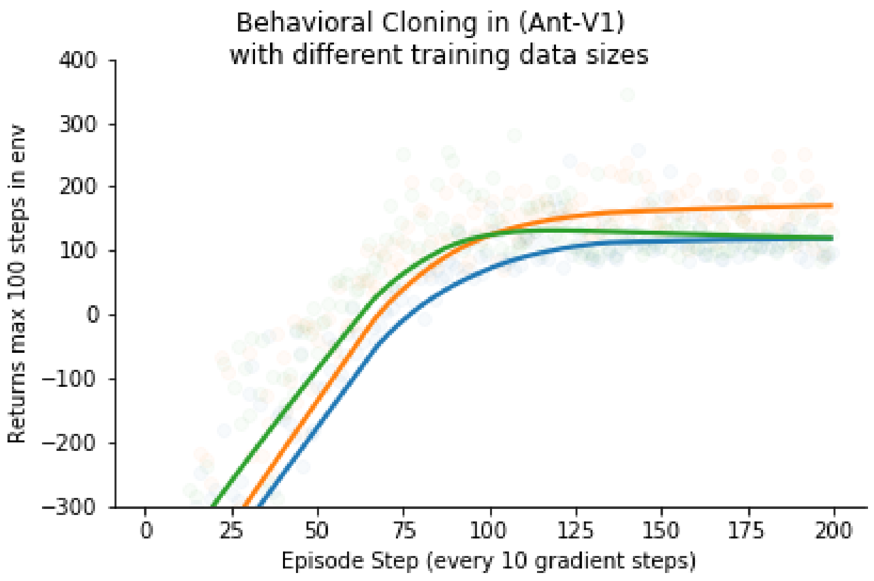
0.1.1 Results

The network achieves somewhat 'comparable' performance for the Cheetah environment, but not for the Ant environment. If I fiddled with learning rate and kept running the gradient steps, cheetah keeps getting closer to expert. Ant seems to plateau.

Env	Mean Return (last 100 episodes)	Std Return (last 100 episodes)
Cheetah	210.1	44.7
Cheetah Expert	284.64	24.33
...	...	...
Ant	222.79	70.51
Ant Expert	412.85	29.93

0.2 Q 3.2 Experimenting with Number of Rollouts

I wanted to try different amounts of training data (ie. number of rollouts). This will likely determine the level at which performance plateua's relative to the expert as there are more and more unique states observed. However, it'll likely take longer to achieve relatively good performance.



ant

0.2.1 Results

I tried 97 (blue), 197 (green), 297 (orange) episodes in the ant-v1. Surprisingly, the size of the training data had a miniminal effect on behavioral clonding's ability to get close to the expert. There is something else that is the issue here (learning rate, quality of training example, etc.)

0.3 Q 4. Dagger

I've decide to implement dagger in the reacher environment. See my HW1_code.ipynb for the code. I've built both the dagger and behavioral cloning inside the same training function. This allowed me to experiment better.

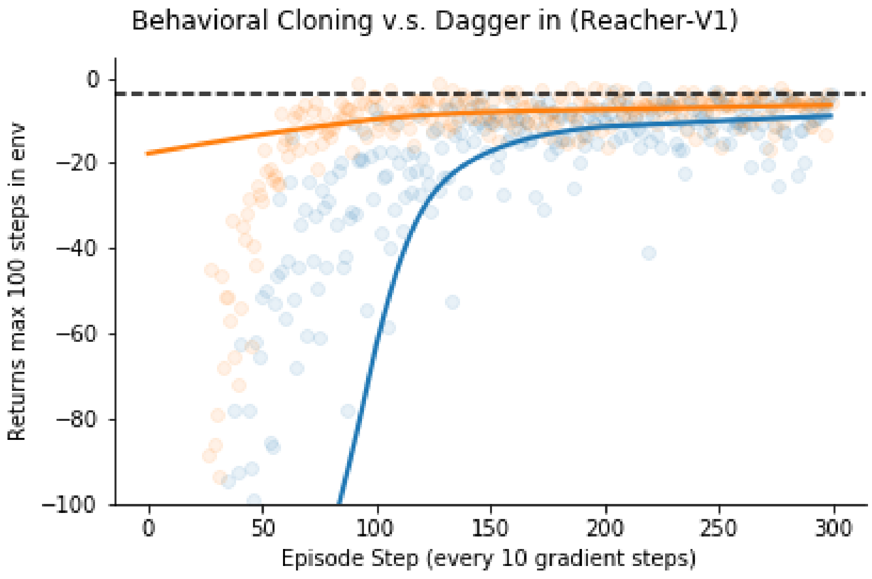
Behavioral Cloning: I ran Reacher with 500 rollouts which generates a 25,000 (s,a) pair dataset.

Interleaving Gym with gradient steps: I kept the same procedure as above: every 10 gradient steps (still with the full dataset), I run an episode in the gym and store the results. (I also tried running a gym episode for each gradient step). After each episode, I label the observations using the expert and append the dataset. The dataset grows with every 10 gradient steps then.

0.3.1 Results:

Dagger does much better for reacher in terms of assymptotic performance. More importantly, it is more efficient. By the end of running the dagger algorithm, the total dataset is 19330 pairs, which is still <25,000 (s,a) pairs for behavioral cloning dataset, yet the performance is much higher for dagger.

Algorithm	Env	Mean Return (last 100 episodes)
behavioral cloning	Reacher	-10.27
dagger	Reacher	-7.01
expert	Reacher	-3.64


Behavioral Cloning v.s. Dagger in (Reacher-V1)
Returns max 100 steps in env
Episode Step (every 10 gradient steps)
0
−20
−40
−60
−80
−100
0
50
100
150
200
250
300

dagger