CS285-HW1

Yuanfang Peng

November 2023

1 Analysis

1.1

we use E_i to denote a random event that expert policy π^* disagree with policy π_θ at time step t . According to the union-bound inequality:

$$Pr[\cup_i E_i] \leq \sum_i Pr[E_i] \tag{1}$$

The expression on the left is actually the probability of the event that policy π_θ makes at least one mistake from the beginning to the end, while the expression on the right is the summation of the probability that policy π_θ makes mistakes at each time step. The latter expression is actually equal to $T\epsilon$ according to hw1.pdf. Then we can derive the following inequity:

$$Pr[\cup_i^t E_i] \leq Pr[\cup_i^T E_i] \leq \sum_i Pr[E_i] \leq T\epsilon \tag{2}$$

It indicates that the probability of the event that policy π_θ makes at least a mistake is less than $T\epsilon$, in other words, the probability of the event that π_θ doesn't make any mistake from time step 1 to time step t. Then we can express $p_{\pi_\theta}(s_t)$ as follows:

$$p_{\pi_\theta}(s_t) = (1 - T\epsilon)p_{\pi^*}(s_t) + T\epsilon p_{mistake} \tag{3}$$

subtract $p_{\pi^*}(s_t)$ from the equation above and get the absolute value:

$$|p_{\pi_\theta}(s_t) - p_{\pi^*}(s_t)| = T\epsilon|p_{\pi^*}(s_t) - p_{mistake}| \tag{4}$$

Apparently:

$$\sum_{s_t} |p_{\pi_\theta}(s_t) - p_{\pi^*}(s_t)| = \sum_{s_t} T\epsilon|p_{\pi^*}(s_t) - p_{mistake}| \leq 2T\epsilon \tag{5}$$

1.2

1.2.1

The reward only depends on the last state, then:

$$J(\pi^*) = \sum_{s_T} p_{\pi^*}(s_T) r(s_T) \tag{6}$$

$$J(\pi_\theta) = \sum_{s_T} p_{\pi_\theta}(s_T) r(s_T) \tag{7}$$

then:

$$J(\pi^*) - J(\pi_\theta) = \sum_{s_T} (p_{\pi^*}(s_T) - p_{\pi_\theta}(s_T)) r(s_T) \tag{8}$$

we can derive the following inequity:

$$|J(\pi^*) - J(\pi_\theta)| = |\sum_{s_T} (p_{\pi^*}(s_T) - p_{\pi_\theta}(s_T)) r(s_T)| \leq \sum_{s_T} |p_{\pi^*}(s_T) - p_{\pi_\theta}(s_T)| r(s_T) \tag{9}$$

in addition:

$$|r(s_t)| \leq R_{max} \tag{10}$$

then:

$$|J(\pi^*) - J(\pi_\theta)| \leq R_{max} \sum_{s_T} |p_{\pi^*}(s_T) - p_{\pi_\theta}(s_T)| \tag{11}$$

according to the conclusion in section 1.1:

$$|J(\pi^*) - J(\pi_\theta)| \leq 2T\epsilon R_{max} \tag{12}$$

$$J(\pi^*) - J(\pi_\theta) = \mathcal{O}(T\epsilon) \tag{13}$$

1.2.2

The inequity also holds for all the time steps, just repeat the equation 12 for all the time steps(T times) and add them together, and we can derive the following inequity:

$$|J(\pi^*) - J(\pi_\theta)| \leq 2T^2\epsilon R_{max} \tag{14}$$

$$J(\pi^*) - J(\pi_\theta) = \mathcal{O}(T^2\epsilon) \tag{15}$$

2 Editing Code

I have filled in all the blanks in the code marked with TODO. In addition to that, I also added some lines to compute the metrics for the expert policy, which has been marked with ADDITIONAL. In terms of how to run the transcript for each part, please refer to cmd.txt.

3 Behavior Cloning

3.1

We run behavior cloning and report its performance on two tasks and also compare its performance with the expert policy. We can see that in the task

policy	mean of return	std deviation of return
ours	4571.25	71.00
expert	4776.44	72.60

Table 1: Comparison of our policy trained from scratch and expert policy on task "Ant-v4". The metrics are mean and standard deviation of reward over 10 rollouts

policy	mean of return	std deviation of return
ours	1050.57	147.68
expert	3716.48	2.71

Table 2: Comparison of our policy trained from scratch and expert policy on task "Hopper-v4". The metrics are mean and standard deviation of reward over 10 rollouts

"Ant-v4" our policy's performance is very close to that of the expert, whereas on the contrary in task "Hopper-v4" our policy achieved less than 30% of the performance of the expert. It indicates that the "Ant-v4" task is easier to learn while the "Hopper-v4" task is harder. Maybe we need more sophisticated design to improve it.

3.2

In this section, we experiment with a set of hyper-parameters that are likely to affect the performance of Behavior Cloning. Fos simplicity, we only experiment on $\{trainingsteps, expertdata\}$

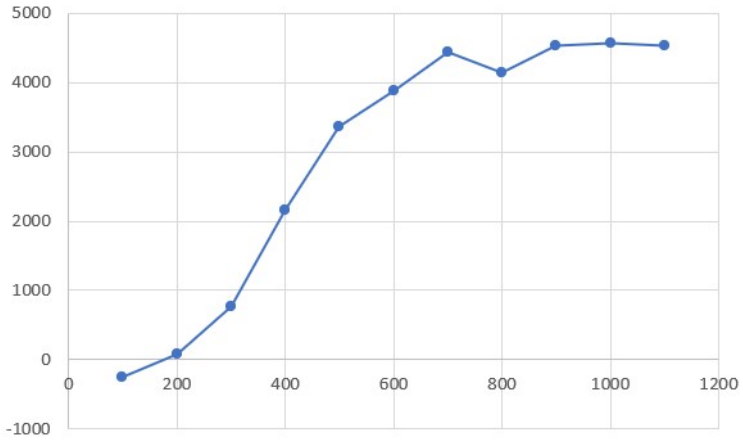

Figure 1: This figure shows how the average reward varies by different training steps. Apparently, the performance increases when the training step increases, finally it levels off because the algorithm has converged.

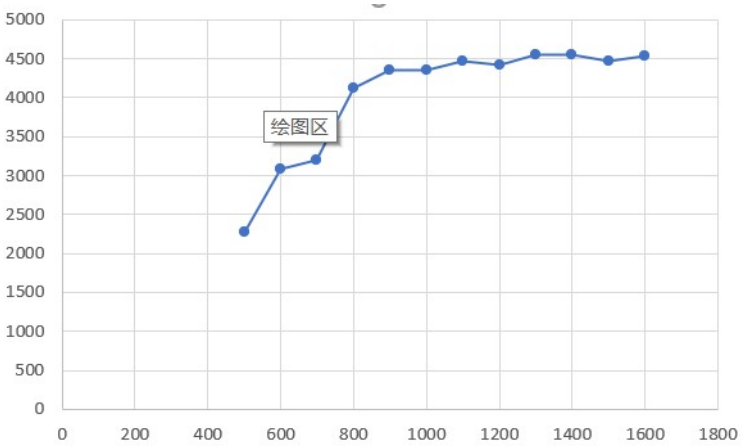

Figure 2: This figure shows how the average reward varies by different number of expert data provided. It increases at first, then when there is enough data provided, it levels off.

4 DAgger

In this section, we show the learning curve of DAgger algorithm. As we expected, after many iterations, the performance of our policy could be very close to the expert policy even for the Hopper-v4 task.

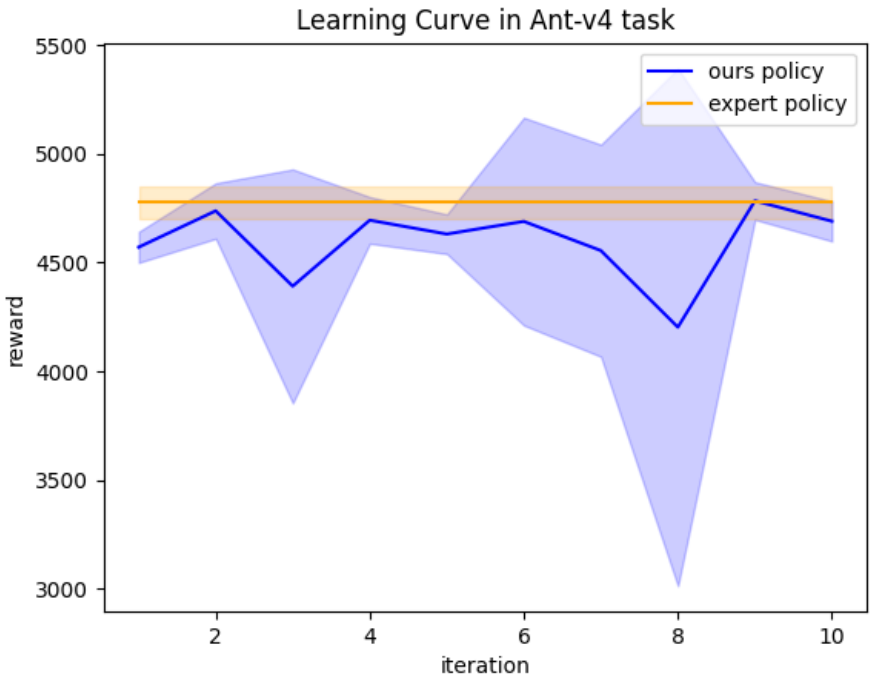

Figure 3: Comparison of our policy and expert policy in Ant-v4 task. Each data point consists of 10 running

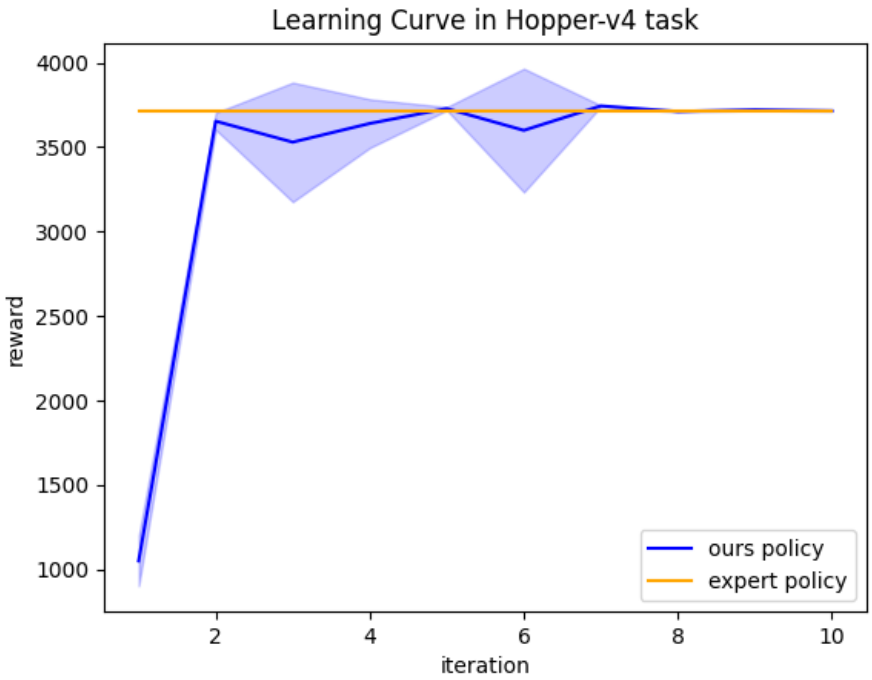

Figure 4: Comparison of our policy and expert policy in Hopper-v4 task. Each data point consists of 10 running