Assignment 1: Imitation Learning

Jules Merigot, Thomas Boudras[1]

[1]Paris Dauphine University - Deep Reinforcement Learning

January 31, 2024

1 Analysis

1.1 We must show that $\sum_{s_t} |p_{\pi_\theta}(s_t) - p_{\pi^*}(s_t)| \leq 2T\epsilon$ using the given information.

First, let's consider an event E_t where π_θ and π^* disagree at time t:

$$E_t = \{\pi_\theta(a_t \neq \pi^*(s_t) \mid s_t)\}$$

The expectation of this event over p_{π^*} is then:

$$\mathbb{E}_{p_{\pi^*}(s_t)} \pi_\theta(a_t \neq \pi^*(s_t) \mid s_t)$$

Using the given equation in the problem statement, we can correctly say that the average of these probabilities over T steps is at most ϵ:

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{p_{\pi^*}(s_t)} \pi_\theta(a_t \neq \pi^*(s_t) \mid s_t) \ \leq \ \epsilon$$

Then, using the union bound inequality, we have:

$$\sum_{t=1}^{T} \mathbb{P}(E_t) = \sum_{t=1}^{T} \mathbb{E}_{p_{\pi^*}(s_t)} \pi_\theta(a_t \neq \pi^*(s_t)|s_t) \ \leq \ T\epsilon$$

Now, the absolute difference in state distributions can be bounded by twice the probability of disagreement. This can be done since the maximum difference can be 2 when there is a complete disagreement:

$$\sum_{s_t} |p_{\pi_\theta}(s_t) - p_{\pi^*}(s_t)| \leq 2\sum_{t=1}^{T} \mathbb{P}(E_t) \ \leq \ 2T\epsilon$$

Therefore, we can conclude that:

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

1.2 Considering the expected return of the learned policy π_θ for a state-dependent reward $r(s_t)$, where we assume the reward is bounded with $|r(s_t)| \leq R_{max}$:

$$J(\pi) \ = \ \sum_{t=1}^{T} \mathbb{E}_{p_{\pi^*}(s_t)} r(s_t)$$

(a) Show that $J(\pi^*) - J(\pi_\theta) \ = \ \mathbb{O}(T\epsilon)$ when the reward only depends on the last state, i.e. $r(s_t) = 0$ for all $t < T$.

First, let's express $J(\pi)$ for π^* and π_θ at time T:

$$J(\pi^*) = \mathbb{E}_{p_{\pi^*}(s_T)} r(s_T) \qquad\qquad J(\pi_\theta) = \mathbb{E}_{p_{\pi_\theta}(s_T)} r(s_T)$$

The difference $J(\pi^*) - J(\pi_\theta)$ can be bounded by the difference in expected rewards under the two distributions, which is affected by ϵ. Given $|r(s_t)| \leq R_{max}$, the maximum difference in rewards at time T is therefore R_{max}. Thus, $J(\pi^*) - J(\pi_\theta)$ can be at most $R_{max} \times 2T\epsilon$ (using result from question 1.1).

Since R_{max} is a constant, we therefore have:

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

(b) Show that $J(\pi^*) - J(\pi_\theta) = O(T^2\epsilon)$ for an arbitrary reward.

Here, rewards depend on the state at each time step. Therefore, the difference $J(\pi^*) - J(\pi_\theta)$ now involves the cumulative difference over T time steps.

Following a similar reasoning as in part (a), but now considering the reward at each time step, this difference can sum up to $T \times R_{max} \times 2T\epsilon$ since there are T terms, each potentially contributing $R_{max} \times 2T\epsilon$ rewards.

Thus, it can be expressed that:

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

2 Editing Code

In the attached code folder...

3 Behavioral Cloning

Part 3.1: Table 1:

Task	**Train Average Return**	**Train standard deviation**	**Evaluation Average Return**	**Evaluation standard deviation**	**Initial Average Return**
Ant	4681.9	30.7	4540	566.1	4681.9
Hopper	3717.5	0.35	1085	332.1	3717.5

Table 1: **Answer 3.1** Results of Behavioral Cloning for different tasks.

Hyperparameters:

- **Episode Length :** 1000 (*)
- **Eval Batch Size :** 10,000 (*10 evaluation episodes are run at each iteration*)
- **Number of Gradient Steps for Policy Training :** 1000
- **Training Batch Size :** 100 (*)
- **Policy :** Neural Network
- **Number of Layers :** 2
- **Width of Each Layer :** 64
- **Learning Rate (lr) :** 5e-3 (*)

The values marked with the symbol (*) indicate the values relevant to question 3.1.

For each iteration, ten episodes are executed. To do this, we set the evaluation batch size to 10000, which is ten times greater than the length of an episode (1000).
During the evaluation in the Hopper environment, we observe a reduction of less than 30% compared with the initial value, while in training mode the value remains the same. Conversely, in the Ant environment, the values are almost identical in evaluation mode to the initial values. However, there is significant variance.

Part 3.2: Figure 1: The below figure uses the same set of hyperparameters as previously stated.

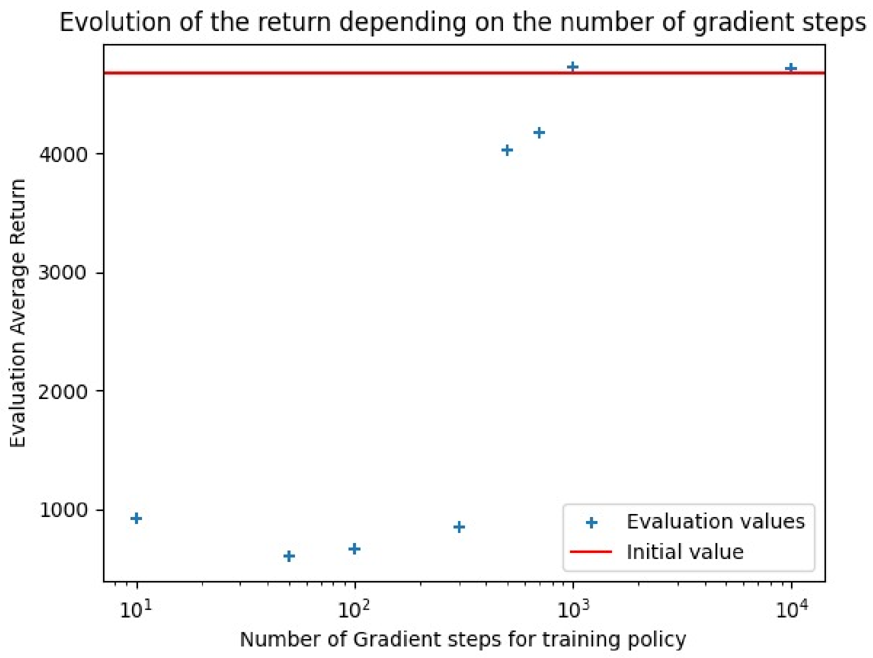

Figure 1: Evolution of the return depending on the number of gradient steps.

We noticed that different training batch sizes affected our results, which makes sense since batch size is known to influence the computational efficiency and convergence speed of a machine learning model. In order to make more informed decisions about our model, we visualized the performance metrics across various batch sizes. This allows us to strike a balance between efficient training and optimal model performance, and thus select the best model.

4 DAgger

Part 4.2: Figure 2: The below figures are made using the same set of hyperparameters as in Question 3.

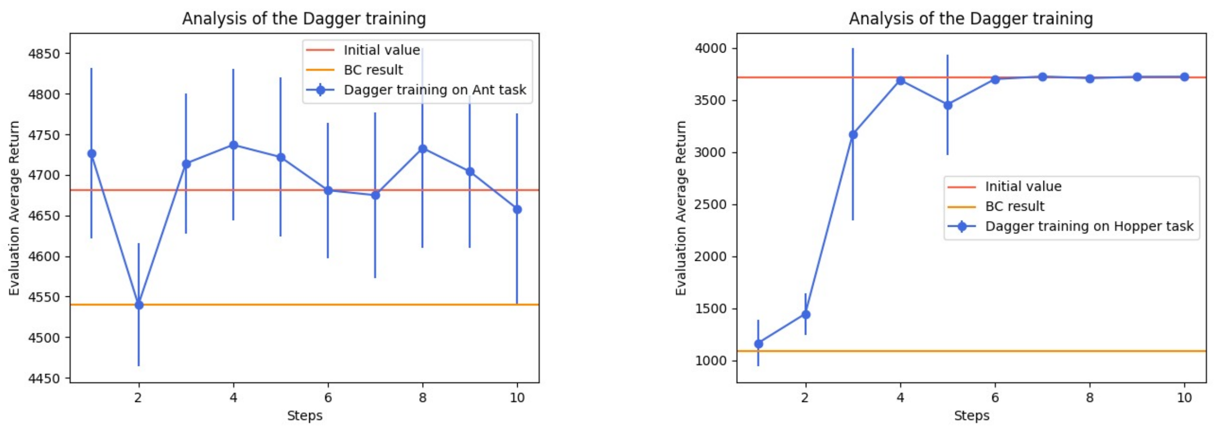

Figure 2: Ant task results (right) and Hopper task results (left).

5 Discussion

1. How much time did you spend on each part of this assignment? We spent 5 hours understanding and filling in the Python code. Then we spent a good hour understanding and analysing the results. Finally, we spent 6 hours producing the curves and the report, much of which was used to understand how TensorBoard works, which we'd never used before.

2. Any additional feedback? Our group would have appreciated it if part of the course had been reserved for introducing the subject and helping us to better understand the structure and functioning of the code.