1 Analysis

CS285 HW1.

No. Date

Consider the problem of imitation learning with discrete MDP with horizon T and expert policy π^*.

We gather expert demonstration from π^* and fit an imitation policy π_θ to these trajectories so that

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

즉, 학습된 정책 π_θ가 전문가 정책 π^*와 불일치할 기대 확률이
전문가 정책 π^*로 생성된 분포 P_{π^*} 하에서 최대 ε가 되도록 한다는 의미이다.

증명을 위해 $P_\pi(s_t)$는 정책 π 하에서 시간 t 시점의 상태를 뜻하며, $P(s)$는 별도로 명시되지 않은 한 전체 시간에 걸친 상태의 평균 분포.

1. Show that $\sum_{s_t} |P_{\pi_\theta}(s_t) - P_{\pi^*}(s_t)| \leq 2T\varepsilon$.

$P_{\pi_\theta}(s_t) = (1-\varepsilon)^t P_{\pi^*}(s_t) + (1-(1-\varepsilon)^t) P_{\text{mistake}}(s_t)$

$P_{\pi_\theta}(s_t) - P_{\pi^*}(s_t) = (1-(1-\varepsilon)^t) |P_{\text{mistake}}(s_t) - P_{\pi^*}(s_t)| \leq \varepsilon t \, |P_{\text{mistake}}(s_t) - P_{\pi^*}(s_t)|$

$(1-\varepsilon)^t \geq 1-\varepsilon t$ 베르누이

두 확률 분포 간 최대 거리 $= 2$ ← Union bound

$\leq 2\varepsilon t$

$\therefore \sum_{s_t} |P_{\pi_\theta}(s_t) - P_{\pi^*}(s_t)| \leq \sum_{s_t} 2\varepsilon t \leq 2\varepsilon T$

PENPIA

No. Date

2. Consider the expected return of the learned policy π_θ for a state-dependent reward $r(s_t)$, where we assume the reward bounded with $|r(s_t)| \le R_{max}$.

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

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

$$J(\pi^*) = \sum_{t=1}^{T} \sum_{s_t} P_{\pi^*}(s_t)\, r(s_t)$$

$$J(\pi_\theta) = \sum_{t=1}^{T} \sum_{s_t} P_{\pi_\theta}(s_t)\, r(s_t)$$

$$J(\pi^*) - J(\pi_\theta) = \sum_{t=1}^{T} \sum_{s_t} \left(P_{\pi^*}(s_t) - P_{\pi_\theta}(s_t)\right) r(s_t) \quad \sum_{t=1}^{T} r(s_t)$$

$\Rightarrow$ reward는 마지막 T에서만 0아님

$$J(\pi^*) - J(\pi_\theta) \le \sum_{s_t} |P_{\pi^*}(s_t) - P_{\pi_\theta}(s_t)| \cdot |R_{max}| = 2T\varepsilon R_{max}$$

$$\therefore J(\pi^*) - J(\pi_\theta) = O(T\varepsilon)$$

(b) Show that $J(\pi^*) - J(\pi_\theta) = O(T^2\varepsilon)$

$$J(\pi^*) - J(\pi_\theta) = \sum_{t=1}^{T} (2\varepsilon t)\, r(s_t) \le \sum_{t=1}^{T} |2\varepsilon t| \cdot |R_{max}| = 2R_{max}\varepsilon \frac{T(T+1)}{2}$$

$$= R_{max}\varepsilon (T^2 + T)$$

$$\therefore J(\pi^*) - J(\pi_\theta) = O(T^2\varepsilon)$$

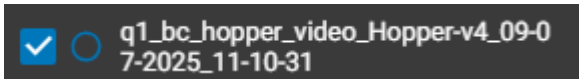
PENPIA

3 Behavioral Cloning

1. Run behavioral cloning (BC) and report results on two tasks: one 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.


```
python cs285/scripts/run_hw1.py --expert_policy_file cs285/policies/experts/Ant.pkl --env_name Ant-v4 --exp_name bc_ant_video --n_iter 1 --expert_data cs285/expert_data/expert_data_Ant-v4.pkl --ep_len 1000 --eval_batch_size 5000 --video_log_freq 1 --n_layers 2 --size 64
```

- Eval_AverageReturn : 1552.6072998046875
- Eval_StdReturn : 292.062255859375
- Eval_MaxReturn : 2147.45068359375
- Eval_MinReturn : 1278.1318359375
- Eval_AverageEpLen : 896.1666666666666
- Train_AverageReturn : 4681.891673935816
- Train_StdReturn : 30.70862278765526
- Train_MaxReturn : 4712.600296723471
- Train_MinReturn : 4651.18305114816
- Train_AverageEpLen : 1000.0
- Training Loss : 0.03868953138589859
- Train_EnvstepsSoFar : 0
- TimeSinceStart : 30.320274114608765
- Initial_DataCollection_AverageReturn : 4681.891673935816

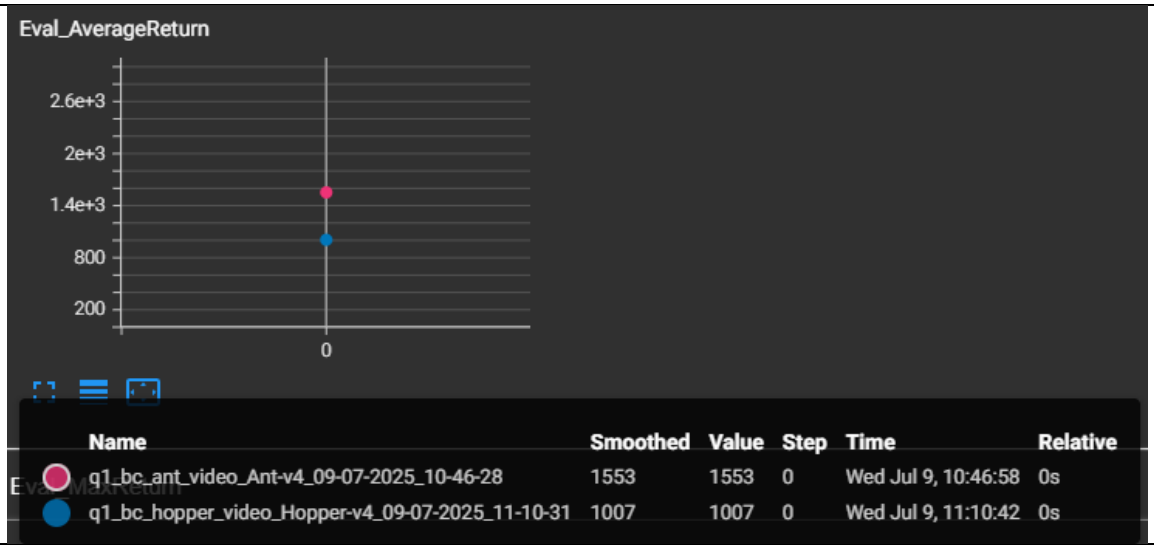



```
python cs285/scripts/run_hw1.py --expert_policy_file cs285/policies/experts/Hopper.pkl --env_name Hopper-v4 --exp_name bc_hopper_video --n_iter 1 --expert_data cs285/expert_data/expert_data_Hopper-v4.pkl --ep_len 1000 --eval_batch_size 5000 --video_log_freq 1 --n_layers 2 --size 64
```

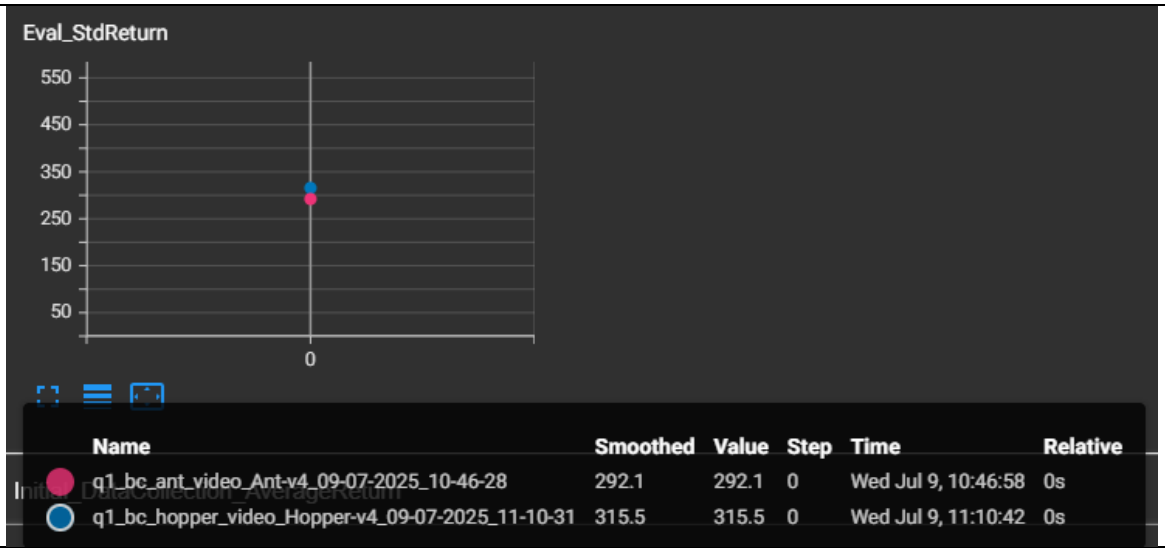
- Eval_AverageReturn : 1006.6049194335938
- Eval_StdReturn : 315.4906005859375
- Eval_MaxReturn : 1974.660400390625
- Eval_MinReturn : 405.4320068359375
- Eval_AverageEpLen : 302.2352941176471
- Train_AverageReturn : 3717.5129936182307

- Train_StdReturn : 0.3530361779417035
- Train_MaxReturn : 3717.8660297961724
- Train_MinReturn : 3717.159957440289
- Train_AverageEpLen : 1000.0
- Training Loss : 0.03620936721563339
- Train_EnvstepsSoFar : 0
- TimeSinceStart : 11.099523305892944
- Initial_DataCollection_AverageReturn : 3717.5129936182307

Eval_AverageReturn



Eval_StdReturn



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. In the caption for the graph, state the hyperparameter and a brief rationale for why you chose it.


```
python cs285/scripts/run_hw1.py --expert_policy_file cs285/policies/experts/Ant.pkl --env_name Ant-v4 --exp_name bc_ant_video --n_iter 1 --expert_data cs285/expert_data/expert_data_Ant-v4.pkl --ep_len 1000 --eval_batch_size 5000 --video_log_freq 1 --n_layers 2 --size 64
```

- Eval_AverageReturn : 1552.6072998046875
- Eval_StdReturn : 292.062255859375
- Eval_MaxReturn : 2147.45068359375
- Eval_MinReturn : 1278.1318359375
- Eval_AverageEpLen : 896.1666666666666
- Train_AverageReturn : 4681.891673935816
- Train_StdReturn : 30.70862278765526
- Train_MaxReturn : 4712.600296723471
- Train_MinReturn : 4651.18305114816
- Train_AverageEpLen : 1000.0
- Training Loss : 0.03868953138589859
- Train_EnvstepsSoFar : 0
- TimeSinceStart : 30.320274114608765
- Initial_DataCollection_AverageReturn : 4681.891673935816


```
... --n_layers 4 --size 128
```

- Eval_AverageReturn : 1277.498291015625
- Eval_StdReturn : 362.19354248046875
- Eval_MaxReturn : 1998.316650390625
- Eval_MinReturn : 870.1470336914062
- Eval_AverageEpLen : 998.5
- Train_AverageReturn : 4681.891673935816
- Train_StdReturn : 30.70862278765526
- Train_MaxReturn : 4712.600296723471
- Train_MinReturn : 4651.18305114816

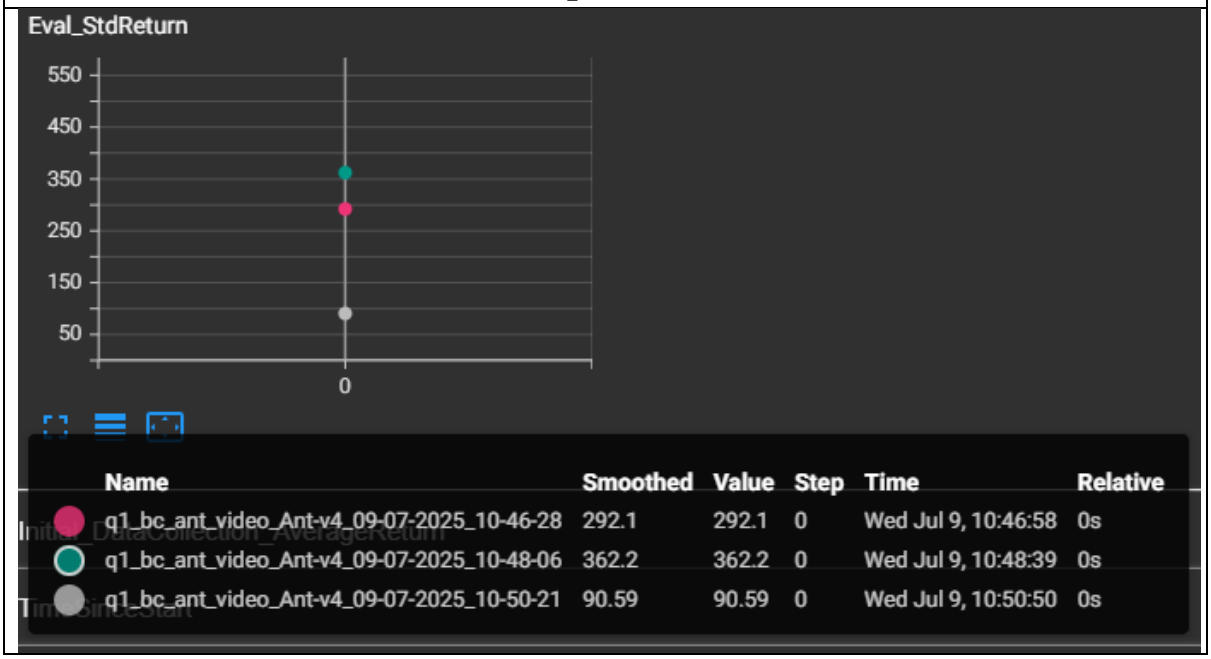
- Train_AverageEpLen : 1000.0
- Training Loss : 0.03484756872057915
- Train_EnvstepsSoFar : 0
- TimeSinceStart : 33.58611559867859
- Initial_DataCollection_AverageReturn : 4681.891673935816


```
... --n_layers 1 --size 32
```

- Eval_AverageReturn : 612.7852783203125
- Eval_StdReturn : 90.59205627441406
- Eval_MaxReturn : 790.5887451171875
- Eval_MinReturn : 546.1014404296875
- Eval_AverageEpLen : 1000.0
- Train_AverageReturn : 4681.891673935816
- Train_StdReturn : 30.70862278765526
- Train_MaxReturn : 4712.600296723471
- Train_MinReturn : 4651.18305114816
- Train_AverageEpLen : 1000.0
- Training Loss : 0.03733008727431297
- Train_EnvstepsSoFar : 0
- TimeSinceStart : 28.81749987602234
- Initial_DataCollection_AverageReturn : 4681.891673935816

Eval_AverageReturn

Eval_StdReturn

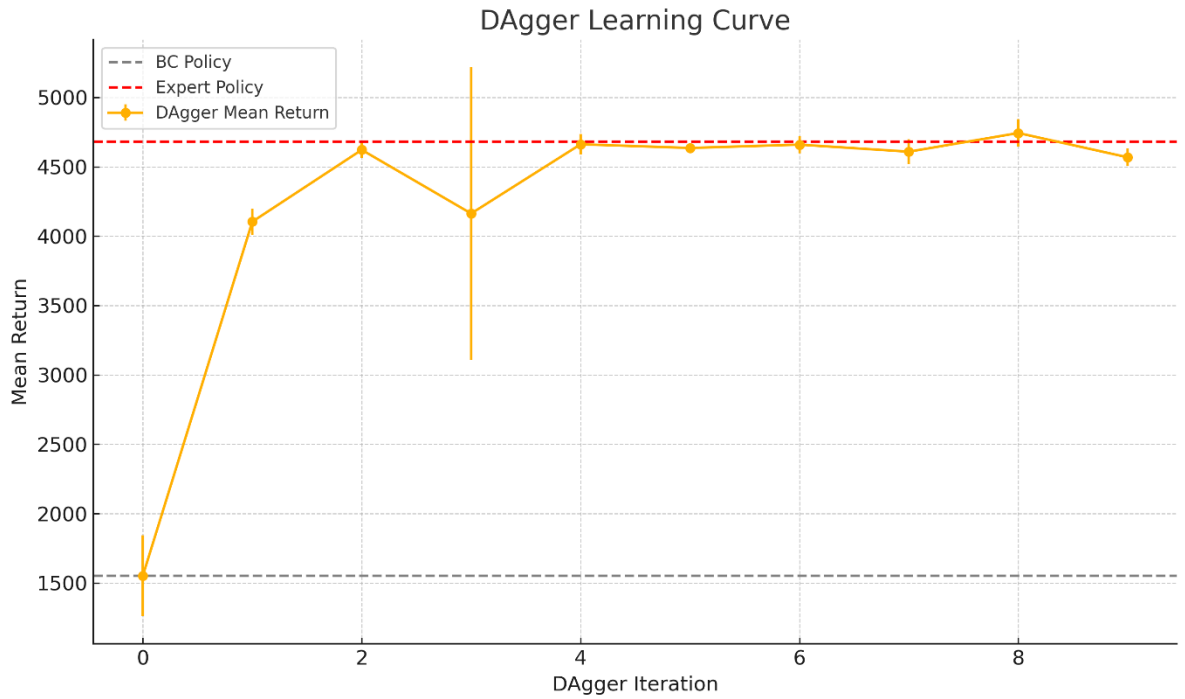


4 Dagger

1. Using the same code, you should be able to run DAgger by modifying the runtime parameters as follows:

```
python cs285/scripts/run_hw1.py --expert_policy_file cs285/policies/experts/Ant.pkl --env_name Ant-v4 --exp_name dagger_ant_video --n_iter 10 --do_dagger --expert_data cs285/expert_data/expert_data_Ant-v4.pkl --ep_len 1000 --eval_batch_size 5000 --video_log_freq 1 --n_layers 2 --size 64
```

2. Run DAgger and report results on the two tasks you tested previously with behavioral cloning. 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).



Environment (Task): Ant-v4

Expert Policy File: Ant.pkl (used to imitate expert behavior)

Learning Method: DAgger (Dataset Aggregation)

- Executed with the --do_dagger flag

Number of Iterations: --n_iter 10 → DAgger was run for 10 iterations

Episode Length: --ep_len 1000 → Each episode consists of 1000 steps

Policy Network Architecture:

- Number of hidden layers: --n_layers 2
- Size of each hidden layer: --size 64

Evaluation Batch Size: --eval_batch_size 5000 → Each evaluation is performed using 5000 steps

5 Discussion

1. How much time did you spend on each part of this assignment.

Analysis	1h
Editing Code	2h
Behavioral Cloning	2h
DAgger	2h

2. Any additional feedback?