Jonathan Pei Date

CS 285 Homework 1: Imitation Learning

1 Analysis

1. Let us define

$$\Pr[\text{mistake}] \coloneqq \Pr[\text{make a mistake at some time step} \leq t].$$

Then, following the same idea as in lecture we have

$$\begin{aligned} p_{\pi_\theta}(s_t) &= (1 - \Pr[\text{mistake}]) \cdot p_{\pi^*}(s_t) + \Pr[\text{mistake}] \cdot p_{\text{mistake}}(s_t) \\ \implies \sum_{s_t} |p_{\pi_\theta}(s_t) - p_{\pi^*}(s_t)| &= \Pr[\text{mistake}] \cdot \sum_{s_t} |p_{\text{mistake}}(s_t) - p_{\pi^*}(s_t)| \\ &\leq 2 \Pr[\text{mistake}] \end{aligned}$$

Now, it suffices to show that $\Pr[\text{mistake}] \leq T\epsilon$:

$$\begin{aligned} \Pr[\text{mistake}] &= \Pr\left(\cup_{0 \leq i \leq t} \text{ first mistake occurs at time step } i\right) \\ &\leq \sum_{i \leq t} \Pr\left(\text{first mistake at time step } i\right) \\ &= \sum_{i \leq t} \Pr\left(\text{matches expert policy until time step } i\right) \\ &\leq \sum_{i \leq t} \sum_{s_i} \Pr[\text{make a mistake at } s_i] \cdot p_{\pi^*}(s_i) \\ &= \sum_{i \leq t} \mathbb{E}_{p_{\pi^*}(s_i)}[\pi_\theta(a_i \neq \pi^*(s_i) \mid s_i)] \\ &\leq \sum_{i \leq T} \mathbb{E}_{p_{\pi^*}(s_i)}[\pi_\theta(a_i \neq \pi^*(s_i) \mid s_i)] \\ &\leq T\epsilon \end{aligned}$$

as desired. Note that we applied the union bound between the 1st and 2nd lines.

2. (a) When the reward only depends on the last state, we have

$$\begin{aligned} J(\pi^*) - J(\pi_\theta) &= \sum_{t=1}^{T} \mathbb{E}_{p_{\pi^*}(s_t)}[r(s_t)] - \sum_{t=1}^{T} \mathbb{E}_{p_{\pi_\theta}(s_t)}[r(s_t)] \\ &= \mathbb{E}_{p_{\pi^*}(s_T)}[r(s_T)] - \mathbb{E}_{p_{\pi_\theta}(s_T)}[r(s_T)] \\ &= \sum_{s_T} r(s_T) \cdot p_{\pi^*}(s_T) - \sum_{s_T} r(s_T) \cdot p_{\pi_\theta}(s_T) \\ &\leq R_{\max} \cdot \sum_{s_T} |p_{\pi_\theta}(s_T) - p_{\pi^*}(s_T)| \\ &\leq 2R_{\max} \cdot T\epsilon \\ &= \mathcal{O}(T\epsilon) \end{aligned}$$

(b) For any arbitrary reward, we have

$$\begin{aligned}
J(\pi^*) - J(\pi_\theta) &= \sum_{t=1}^{T} \mathbb{E}_{p_{\pi^*}(s_t)}[r(s_t)] - \sum_{t=1}^{T} \mathbb{E}_{p_{\pi_\theta}(s_t)}[r(s_t)] \\
&= \sum_{t=1}^{T} \sum_{s_t} r(s_t) \cdot (p_{\pi^*}(s_t) - p_{\pi_\theta}(s_t)) \\
&\leq R_{\max} \sum_{t=1}^{T} \sum_{s_t} |p_{\pi_\theta}(s_t) - p_{\pi^*}(s_t)| \\
&\leq R_{\max} \sum_{t=1}^{T} 2T\epsilon \\
&= 2R_{\max} \cdot T^2\epsilon \\
&= \mathcal{O}(T^2\epsilon)
\end{aligned}$$

2 Behavioral Cloning

1. I report the mean and standard deviation of my policy's return (over multiple rollouts) on the Ant and Walker2d tasks:

Task	`Eval_AverageReturn`	`Eval_StdReturn`
Ant	3876.619	1213.65
Walker2d	651.389	909.57

Table 1: **Evaluation Mean and Standard Deviation of my policy's return over multiple rollouts, on the Ant and Walker2d tasks.** I set `seed` to 4, `ep_len` to 1000, and `eval_batch_size` to 10000. All other hyperparameters (e.g. model architecture, learning rate, etc.) are identical and the default values. We observe very different results for the Ant and HalfCheetah tasks, even though the training configurations used were the exact same. In particular, our policy was able to achieve 81.2% of the performance of the expert on the Ant task, and only 12.1% on the Walker2d task.

2. I experimented with varying the number of hidden layers in the MLP model, on the Walker2d task. A graph is provided below:

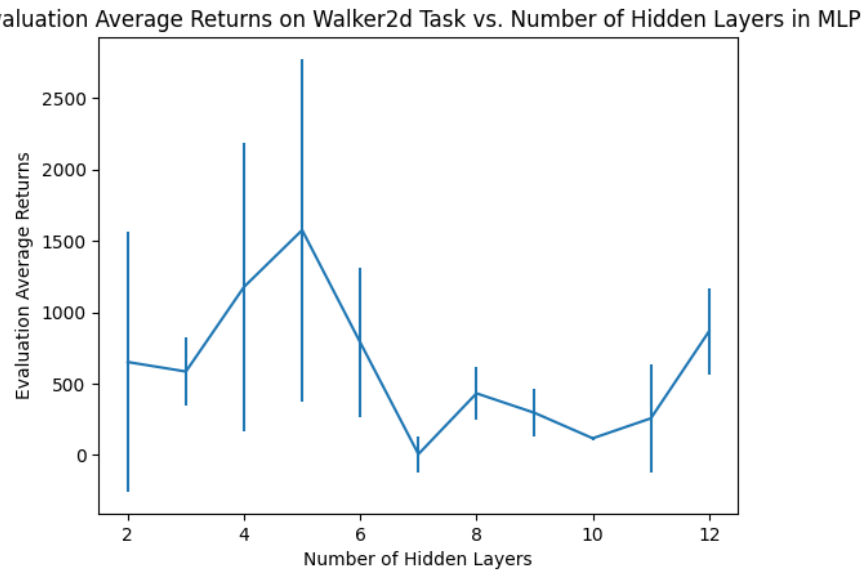

Figure 1: **Evaluation Average Returns on the Walker2d task vs. Number of MLP Hidden Layers.** I varied the hidden layer from 2 to 12, and kept all other training configurations the same as in part (1). We can see that the agent's performance increases until it peaks at around 5 hidden layers, after which it mostly decreases until 12 layers. I wanted to try adjusting the number of hidden layers because deeper neural networks are capable of learning more complex patterns, until a certain point where they begin to overfit. My experiment (roughly) exhibits this behavior.

3 DAgger

2. After running DAgger on the Ant and Walker2d tasks, I present the following two graphs:

Figure 2: **Average Returns on Ant and Walker2d Tasks vs. Number of DAgger Iterations.** I ran the agent on the Ant and Walker2d tasks, with the same configurations as described in the previous section. We see that DAgger substantially outperforms behavioral cloning, especially in Walker2d.

4 Discussion

1. I spent 2 hours on Q1, 6 hours on Q3, and 2 hours on Q4.

2. Nope.