Assignment 1

Dylan Slack

Part 1 Problems

1

If the initial distribution is a stationary distribution, then $p(s_0)P = p(s_0)$ so the distribution over states remains $p(s_0)$. Let R be the average one step reward given P and $p(s_0)$. Because $p(s_0)$ is a stationary distribution, the average one step reward R is average one step reward in each state visited until the process terminates.

Then, for a T-step finite horizon, the expected reward is $\sum_{t=0}^{T-1} R = TR$. For a discounted horizon, the expected reward is $\sum_{t=0}^{\infty} \gamma^t R = \frac{R}{1-\gamma}$, and for an infinite horizon this is $\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} R = \lim_{T\to\infty} \frac{1}{T} TR = \lim_{T\to\infty} R = R$. Thus, its clear that these values are related by a multiplicative constant on R. In a T-step finite horizon, if we set $T = \frac{1}{1-\gamma}$ we would get the same multiplicative constant. Though, I'm unsure what this would mean when T results in a non-integer value.

2

The stationary distribution is then the distribution always in the absorbing state because the transition always results in this same distribution per the policy. This would be the distribution over the states with 100% chance of being in the absorbing state.

3

Supposing that we have some MDP $\mathcal{M}_1$ with a discounted horizon to start and are interested in converting this to an MDP $\mathcal{M}_2$ of episodic horizon, we first start by adding an absorbing state such that once in the absorbing state, taking any action results in the absorbing state. Next, we set the rewards (from states to the absorbing state and from the absorbing state to the absorbing state to 0). To make $\mathcal{M}_2$ an equivalent episodic MDP, we finally update the transition probabilities to the absorbing state to $1-\gamma$ where γ is the discount factor from $\mathcal{M}_1$. We additionally normalize the non-absorbing state transitions to make sure the transition probability ends up with 1. We see in this updated scenario that heavily discounted MDP's will produce traversals that value maximizing immediate rewards because the expected number of state traversals before transitioning to the absorbing state is lower.

Part 2 Implementation Section

Results for behavior cloning

I considered both the half cheetah and walker2d environments for evaluation. I considered the default parameters for training and 5 roll outs of evaluation for assessment between both the environments in order to facilitate a fair comparison. The results are given in table 1.

	Half Cheetah	Walker2D
Mean Training	4206	5567
Std Training	83	9
Mean Evaluation	2891	626
Std Evaluation	155	557

Table 1: Using default implementation parameters comparison in reward between behavior cloning in half cheetah and walker 2d.

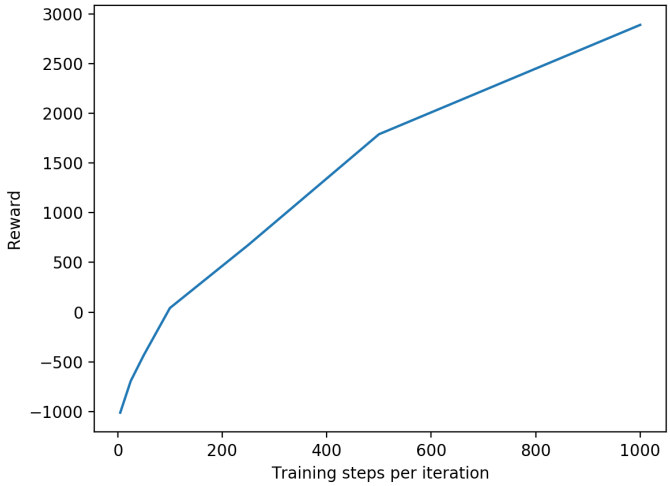

Figure 1: Training steps per iteration in comparison to cumulative reward for behavior cloning on half cheetah task.

We see that on the half cheetah task, this strategy produces a much higher proportion of the expert policy than the walker example.

Next, we look at an example where I vary the number of training steps per iteration using the default parameter half cheetah policy from above. This results to this experiment are given in figure 1. We see that a longer number of iterations produces better cumulative reward. I considered this variation because it makes sense that a longer optimized policy could produce better results. These results reflect this intuition.

DAgger

Next, I considered DAgger in comparison to behavior cloning on the Walker2D task. I observed noticeable improvement using DAgger on this task. I set the neural network configuration the same between both of these algorithms — just to their default configurations. Next, I considered 5 roll outs (episode length 1000, evaluation batch size 5000) for each iteration of DAgger and compared this to the expert policy and the behavior cloning policy. The results are given in 2. We observe that DAgger is able to generate better results on this task than the behavior cloning policy and greatly improves its variable ability to do so as the number of iterations progresses. Interestingly, DAgger on the second iteration produces fairly good results but then diverges away from this behavior and produces much poorer results until it reconverges back.

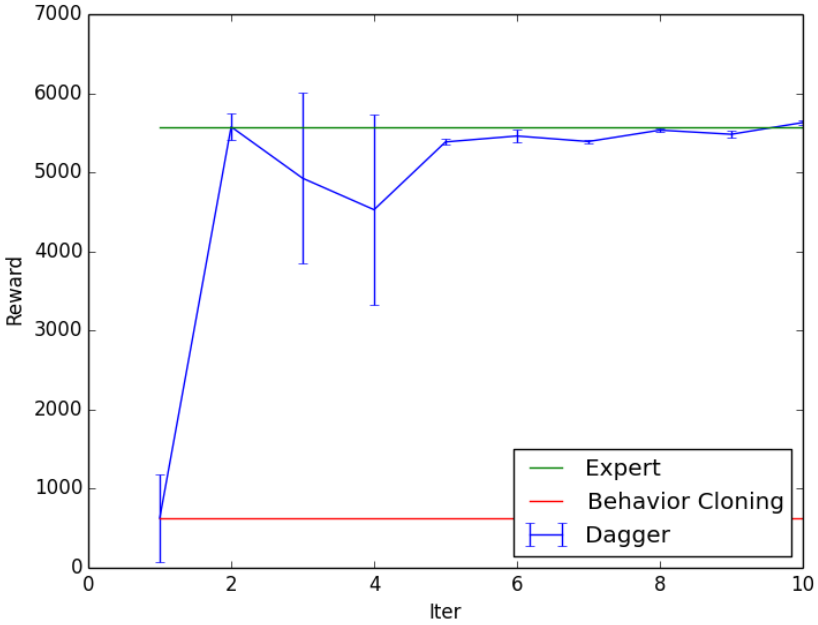

Figure 2: Comparison of DAgger to behavior cloning and expert policy on the walker 2d. Error bars represent the standard deviation.

Statement of collaboration: I only worked on this by myself.