## Instructions

This homework is designed to follow up on the lecture about Deep Imitation Learning. For this assignment, you will need to know about the imitation learning algorithms we talked about in the class. If you have not already, we propose you brush up on the lecture notes. Additionally, this homework will introduce DAgger, an important imitation learning algorithm, and you will implement part of it as part of the assignment.

**You are ALLOWED to discuss this homework with your classmates at the level of general solution strategies or tips. However, any work that you submit must be entirely your own, which includes your own unique code and your own unique written answers.**

**You are NOT ALLOWED to use any large language models (LLMs) or other AI-assisted tools.**

If you find errors in the homework assignment or have public questions, please post in the Campuswire “#questions-hw” channel or ask during instructor/TA office hours.

## Points

| Q1 | DAgger Implementation |  | 10 pts |
| --- | --- | --- | --- |
| Q2 | Plots: Expert Queries vs. Success Rate |  | 5 pts |
| Q3 | Improving DAgger |  | 5 pts |
| Q4 | Improving DAgger and Compare |  | 5 pts |
|  |  |  | **25 pts total** |

## 

## Coding Questions

Code: <https://drive.google.com/file/d/190PHgnIJdeKL_J_JJcKB765nJItAHGUM/view?usp=sharing>

You are only required to modify the code provided in the *policy/dagger\_template.py* file. The virtual env can be set up using *conda\_env.yml* and instructions for running the code have been included in *instructions.md*.

**Note: Please create a new Conda environment since *particle-envs* has been modified.**

## DAgger

In this homework, we introduce an important imitation learning algorithm called DAgger. While Behavior Cloning works well with perfect demonstrations, it struggles when the learned policy encounters states not seen during training, causing small mistakes to accumulate. DAgger addresses this by allowing the learned policy to interact with the environment, and gather action labels from the expert for the states it encounters. This feedback loop helps the policy learn to recover from its own errors, making it more robust. This method is quite useful especially when querying the expert is expensive, and thus we want to learn a policy that is almost as good as the expert without the high number of queries to it.

In this homework, we have provided you with an environment that is hard to learn directly. Thankfully, we have access to an expert in this environment. **In this homework, your task will be to utilize *DAgger* to learn a deep neural network policy that performs well on this task.**

**Resources:**

DAgger paper: <https://proceedings.mlr.press/v15/ross11a/ross11a.pdf>

Video: <https://www.youtube.com/watch?v=awfrsjYnJmw>

### Environment

The environment we will use in this homeworks follows the OpenAI gym API, which you can learn more about at <https://github.com/openai/gym#api>. It is a goal-reaching environment where the agent is spawned at a start location and is tasked with reaching a goal location.

This is the same environment as Assignment 1 with a minor change. Previously, the episode would terminate at the first instance of the agent reaching the goal. Now, each episode always runs for a fixed length of 300 time steps and a episode is considered successful only if the last step of the episode is at the goal. This has been done to make the task slightly harder since the agent must now learn to reach the goal and stop there as opposed to overshooting the goal in the previous version.

Further, since we want to learn a policy using DAgger, we have access to an expert prediction for any state the environment is currently on, which can be retrieved by the get\_expert\_action() function call. **Note: get\_expert\_action() does not take any arguments, thus you must be careful to call it right after you have called .reset() or .step() on the environment to get the associated expert action.**

**Deliverables:**

* Within PDF write-up, any written answers and plots.
* Within the compressed ZIP folder, the *policy/dagger\_template.py* file. Compress all files into a zipped folder. Please name your file as *<net\_ID>\_assignment2.zip*

### Question 1

Download the code folder, with every file associated, from here <https://drive.google.com/file/d/190PHgnIJdeKL_J_JJcKB765nJItAHGUM/view?usp=sharing>

Complete the code template provided in dagger\_template.py, with the right code in every TODO section, to implement DAgger. Attach the completed file in your submission.

### Question 2

Create a plot with the number of expert queries on the X-axis, and the success rate of the imitation model on the Y-axis. Elaborate if you see any clear trends here. (Hint: in the env, the variable expert\_calls counts the number of expert queries.)

### Question 3

Could you potentially improve on the number of queries to the expert made by the DAgger algorithm? Think about when querying the expert may be redundant.

Question 4

Try implementing your answer from Question 3, and generate a *query vs success rate* plot similar to Question 2 for this implementation. Compare this plot with your answer from Q2. Is there a clear improvement?