CS224R Spring 2025 Homework 1
Imitation Learning

Due 4/18/2025

SUNet ID: agliang
Name: Agnes Liang
Collaborators:

By turning in this assignment, I agree by the Stanford honor code and declare that all of this is my own work.

Overview

Goals: In this assignment you will experiment with two common methods of imitation learning: behavior cloning and DAgger. You will then have the opportunity to train and tune your agents in a variety of environments in the MuJoCo physics simulator (`https://mujoco.org/`). In lieu of a human demonstrator, we will provide demonstrations from a pre-trained expert policy.

1. Implement and train an imitation learning agent that can learn from expert data.
2. Analyze the shortcomings of behavior cloning and how DAgger can address some of these issues.
3. Experiment with hyperparameters and explore how they affect performance.

Submitting the PDF: Make a PDF report containing: Table 1 for a table of results from Question 1.2, Figure 1 for Question 1.3, and Figure 2 with results from Question 2.2 as well as the requested brief explanations.

Submitting the Code and Experiment Runs: In order to turn in your code and experiment logs, create a folder that contains the following:

- A folder named data with experiments for both the behavioral cloning (sub-problem 1, not sub-problem 2) exercise and the DAgger exercise. Note that you can include multiple runs per exercise if you'd like, but you must include at least one run (of any task/environment) per exercise. These folders can be copied directly from the `cs224r/data` folder into this new folder. **Important: Disable video logging for the runs that you submit, otherwise the files size will be too large! You can do this by setting the flag –video log freq -1**
- A README file that lists the commands (with clear hyperparameters) that we need in order to run the code and produce the numbers that are in your figures/tables. The following command is an example:

```
To generate the numbers for Question 1 Subproblem 1, run:
python cs224r/scripts/run_hw1.py \
--expert_policy_file cs224r/policies/experts/Ant.pkl \
--env_name Ant-v4 --exp_name bc_ant --n_iter 1 \
--expert_data cs224r/expert_data/expert_data_Ant-v4.pkl
```

- The cs224r folder with all the .py files, with the same names and directory structure as the original homework repository.

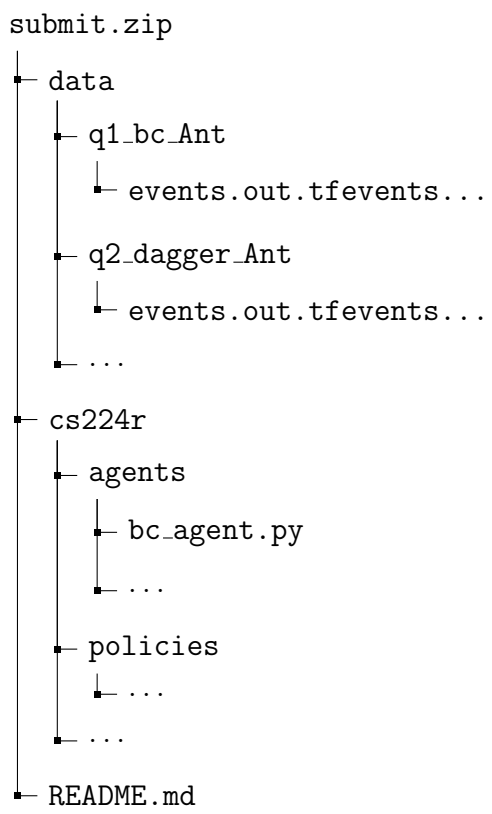
Gradescope: Submit both the PDF and the code and experiment runs in the appropriate assignment on Gradescope. An autograder will be provided to evaluate the performance of your policies from the generated tensorboard files. **Note: the autograder score will only consist of 50% of your homework grade. The consistency of the code and tensorboard files will be checked after submission.**

Note: once you have the files for submission e.g. in a folder named `submit`, run the command `zip -r submit.zip submit/` to generate the zip file. For an example of what the zip should look like, see the following page. In the past, we've found programs like WinRAR seem to be incompatible with Gradescope.

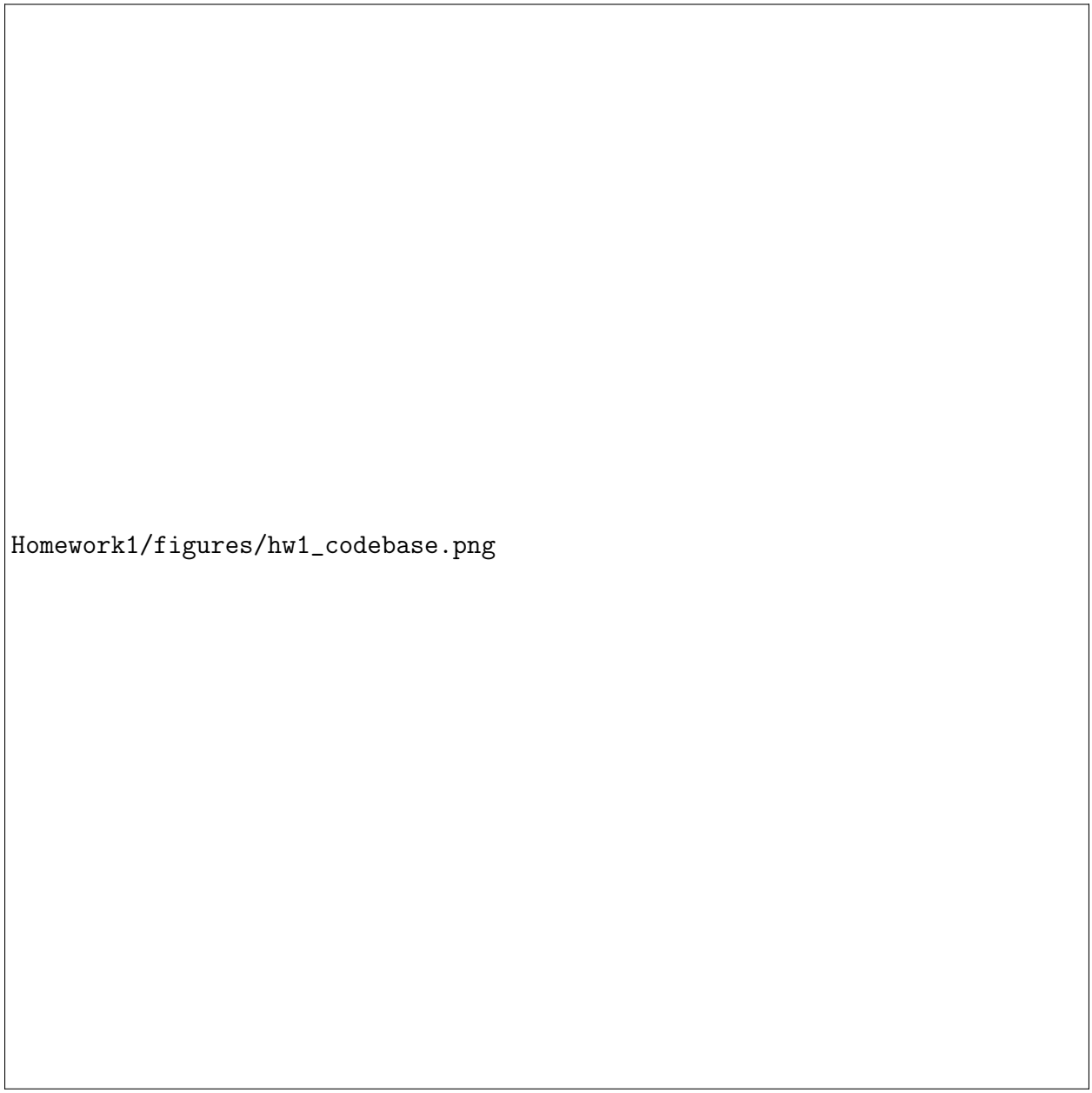
Use of AI Tools (e.g. ChatGPT, Cursor): For the sake of deeper understanding on implementing imitation learning methods, assistance from generative models to write code for this homework is prohibited. See the course website for more details.

Sample File Submission

As an example, the unzipped version of your submission should result in the following file structure. **Make sure that the submit.zip file is below 15MB and that the experiment log folders include the prefix `q1_` and `q2_`.**



Codebase



We have provided you with starter code for this homework. A simplistic diagram of its structure is shown above. In the `agents` directory, you will define your main behavior cloning agent. This agent will contain contain references to its policy (or actor) and replay buffer. This agent will be fed into the BC trainer, which will contain its training loop.

We recommend that you read the files in the following order. Your task is to fill in the blanks labeled `TODO`.

- `scripts/run_hw1.py (read-only file)`

- infrastructure/bc_trainer.py
- agents/bc_agent.py (read-only file)
- policies/MLP_policy.py
- infrastructure/replay_buffer.py
- infrastructure/utils.py
- infrastructure/pytorch_util.py

Problem 1: Behavior Cloning

1. Run behavioral cloning (BC) and report results on two tasks: the Ant environment, 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. For your reference, the other environments with expert data include Hopper, Walker2d, and HalfCheetah. A policy that achieves greater than 30% of the expert on the Ant task will receive full credit on the autograder. Here is how you can run the Ant task:

```
python cs224r/scripts/run_hw1.py \
    --expert_policy_file cs224r/policies/experts/Ant.pkl \
    --env_name Ant-v4 --exp_name bc_ant --n_iter 1 \
    --expert_data cs224r/expert_data/expert_data_Ant-v4.pkl \
    --video_log_freq -1
```

   When providing results, report the mean and standard deviation of your policy's return over multiple rollouts in a table, and state which task was used. When comparing one that is working versus one that is not working, be sure to set up a fair comparison in terms of network size, amount of data, and number of training iterations.

   Note: To report the mean and standard deviation, your eval batch size should be greater than `ep_len`, so that you're collecting multiple rollouts when evaluating the performance of your trained policy. For example, if `ep_len` is 1000 and `eval_batch_size` is 5000, then you'll be collecting approximately 5 trajectories (maybe more if any of them terminate early), and the logged `Eval_AverageReturn` and `Eval_StdReturn` represents the mean and standard deviation of your policy over these 5 rollouts.

   Tip: To generate videos of the policy, remove the flag `--video_log_freq -1`. However, this is slower, and so you will want to keep this flag on while debugging.

Problem #1.1					
Eval_batch_size = 5000					
Environment	**Expert Mean Return**	**Expert StdDev**	**BC Agent Mean Return**	**BC Agent StdDev**	**% of Expert Performance**
Ant-v4	4713.65	12.2	4596.11	110.28	97.50%
Walker2d-v4	5566.85	9.24	481.92	493.87	8.70%

Figure 1: Problem 1 Table

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. State the hyperparameter and a brief one or two sentence rationale for why you chose it. The plot should contain clearly labeled axes.

Include your chosen hyperparameter, plot, and rationale here.

Problem #1.2			
Environment = Ant-v4			
Hyperparameter : Number of Training Steps			
Rationale:I chose to change the number of training steps to see how it affects the agent's learning from expert data. With too few steps, the agent doesn't learn much and performs poorly. As I increased the number of steps, the agent got better at copying the expert. By 1000 steps, it was nearly as good as the expert. This shows that giving the agent more time to learn during training helps improve its performance.			
# of Training Steps	**BC Agent Mean Return**	**BC Agent StdDev**	**% of Expert Performance**
1	-129.93	91.85	-2.97
10	-499.93	883.25	10.60%
100	-128.79	744.37	97.30%
1000	4615.41	0	97.90%
10000	4689.15	0	99.50%

Figure 2: Problem 1 Table

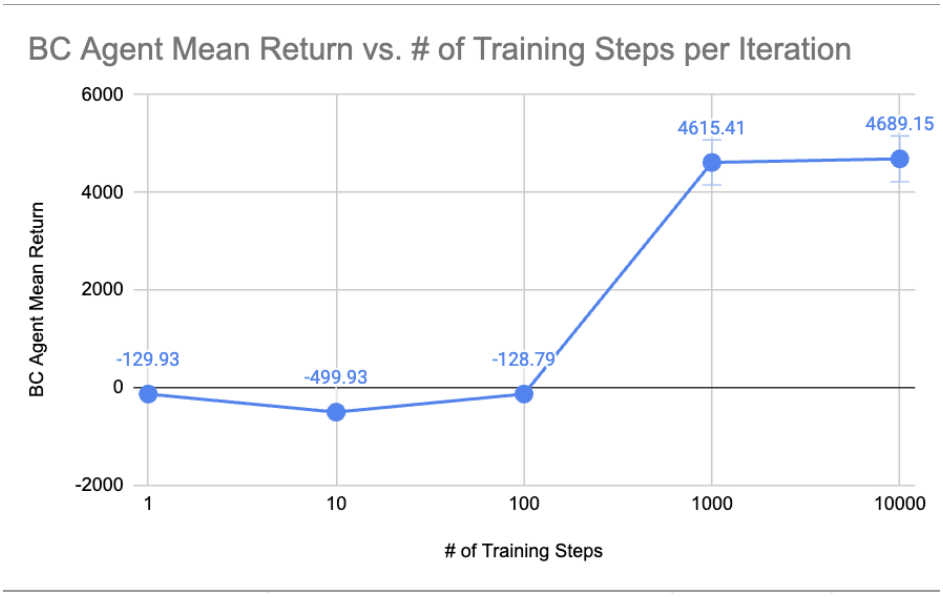


Figure 3: Problem 1 Table

Problem 2: DAgger

1. Once you've filled in all of the TODO commands, you should be able to run DAgger.

```
python cs224r/scripts/run_hw1.py \
  --expert_policy_file cs224r/policies/experts/Ant.pkl \
  --env_name Ant-v4 --exp_name dagger_ant --n_iter 10 \
  --do_dagger \
  --expert_data cs224r/expert_data/expert_data_Ant-v4.pkl \
  --video_log_freq -1
```

2. Run DAgger and report results on the two tasks you tested previously with behavioral cloning (i.e., Ant + another environment). Report your results in the form of a learning curve, plotting the number of DAgger iterations on the x-axis versus the policy's mean return on the y-axis, with error bars to show the standard deviation. Include the performance of the expert policy and your behavioral cloning agent on the same plot. State which task you used, and any details regarding network architecture, amount of data, etc. (as in the previous section).

   Include the plots of the two tasks here.

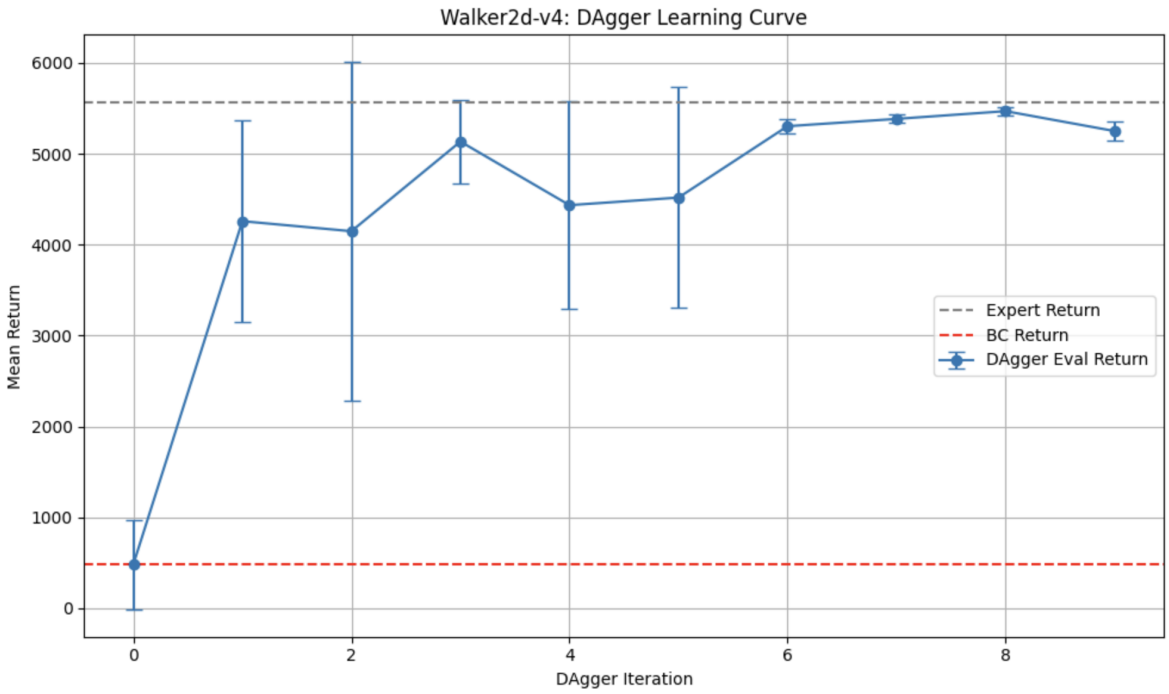


Figure 4: Problem 1 Table

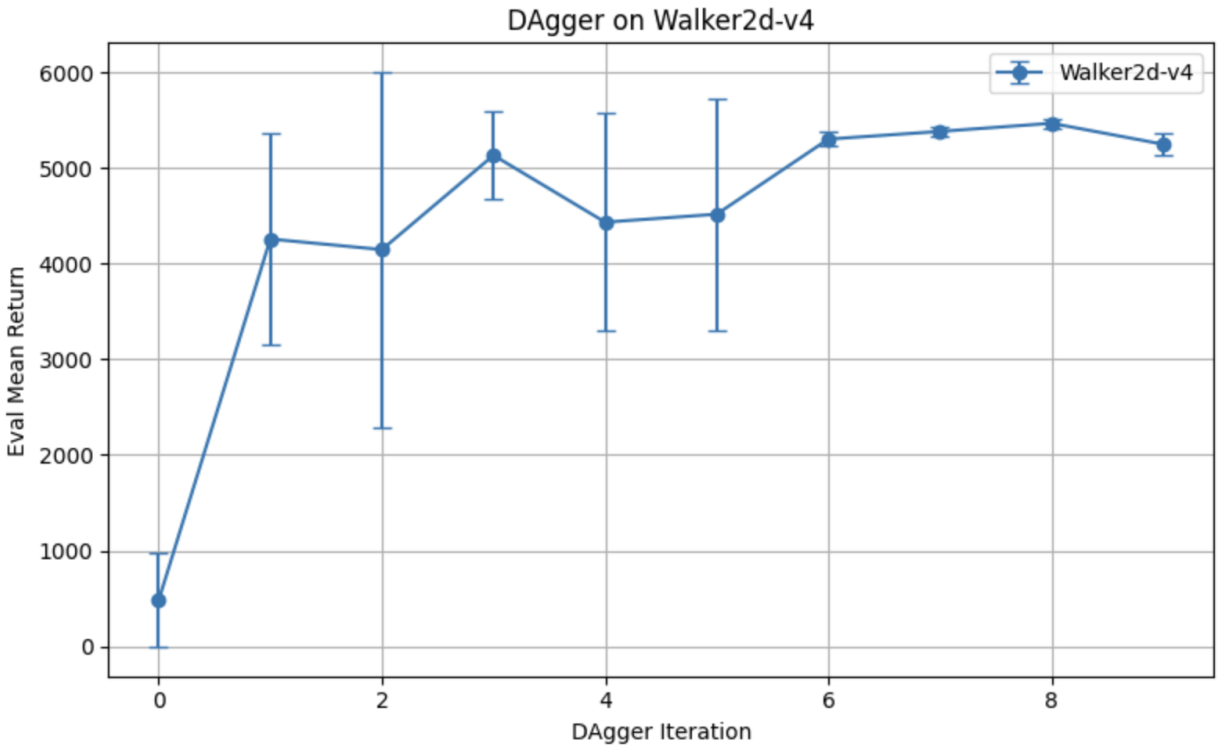


Figure 5: Problem 1 Table

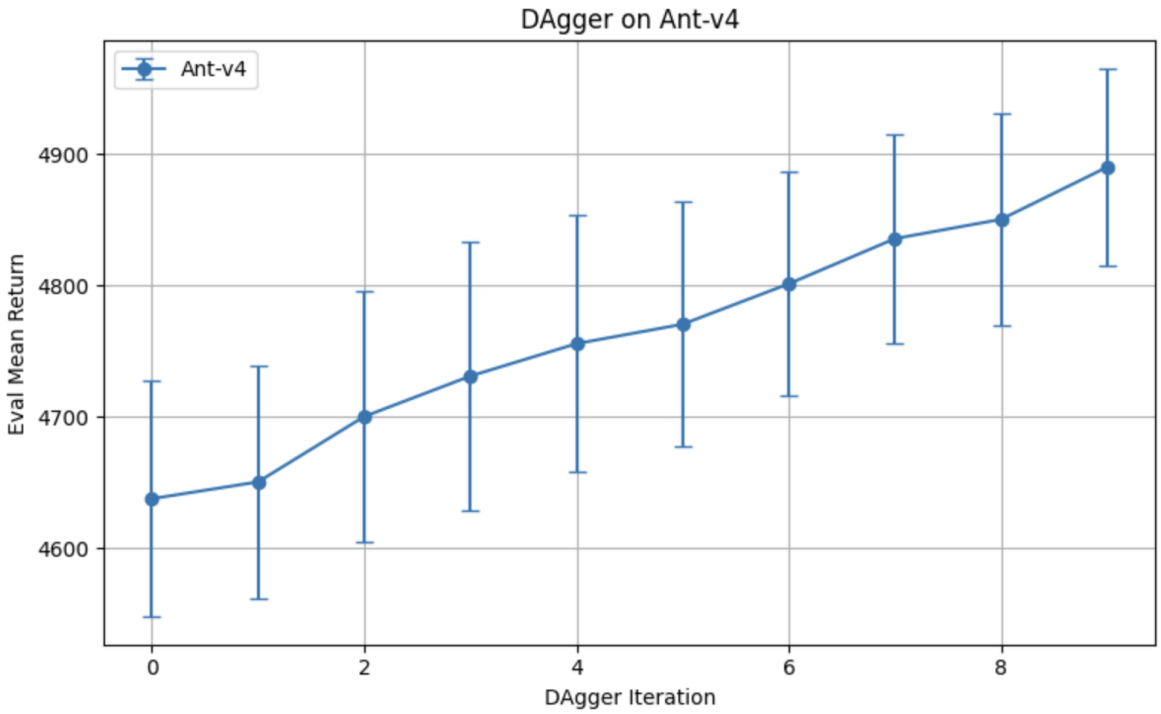


Figure 6: Problem 1 Table

We ran DAgger on two MuJoCo tasks: Ant-v4 and Walker2d-v4. For both, we used expert demonstration data to initialize the policy and trained for 10 DAgger iterations, collecting 5000 steps per iteration with an episode length of 1000. The policy was a 2-layer MLP with 64 hidden units, ReLU activations, and a Gaussian action output. We evaluated each iteration using a batch size of 5000, reporting the mean and standard deviation of returns across rollouts.