AAI4160 Homework 1: Imitation Learning

[Your name]
[Your student ID]

1 Introduction

The goal of this assignment is to make you familiar with **(1) Yonsei AI Teaching Cluster (VESSL AI)**, **(2) Reinforcement Learning Environments**, and **(3) Imitation Learning Algorithms**, including behavioral cloning (BC) and DAgger. Here is the link to this homework report template in Overleaf.

2 (Optional) VESSL AI

If you have your own GPUs, you can skip this section.

If you need a GPU for this assignment, we provide VESSL AI (AI-LEC-1 group), a GPU cluster for lectures in the Yonsei AI department. You can use an RTX3090 GPU for 300 hours per month. Here is the instruction about how to launch and use a workspace (i.e. a virtual machine with one GPU assigned to you) for this homework.

Warnings:

- Always stop the workspace when not in use to save the computing resources.
- The **VESSL Workspace** provides a maximum allocation of 72 hours of runtime. Exceeding this limit will automatically stop your workspace. Monitor your usage regularly to manage your available hours, and restart the workspace if it has been automatically stopped.

3 Play with Reinforcement Learning Environments

We provide a tutorial (`GymTutorial.ipynb`) inside the homework material to help you familiarize yourself to Reinforcement Learning (RL) environments. We encourage you to review and run this tutorial to understand how to interact with gym environments.

After completing the tutorial, please answer the following questions to demonstrate your understanding of the basic principles of gym environments.

3.1 Questions:

1. Describe the role of the `step()` function. What kind of information does it return?
   Answer: Include your answer here.

2. Describe the role of the `reset()` function in a gym environment. What is the return value of this function?
   Answer: Include your answer here.

3. How can you figure out if some gym environment has a discrete action space or continuous action space?
   Answer: **Include your answer here.**

4 Behavioral Cloning

4.1 Implement Behavioral Cloning

Your task is to fill in sections marked with `TODO` in the code. In particular, see the following files:

- `aai4160/infrastructure/bc_trainer.py`, except for `do_relabel_with_expert` function, which is for the next section, DAgger.
- `aai4160/policies/MLP_policy.py`
- `aai4160/infrastructure/replay_buffer.py`
- `aai4160/infrastructure/utils.py`
- `aai4160/infrastructure/pytorch_util.py`

Run behavioral cloning (BC) and report results on **two tasks**: (1) the Ant environment (`Ant-v4`), where a behavioral cloning agent should achieve at least 30% of the performance of the expert, and (2) any one environment among `Walker2d-v4`, `HalfCheetah-v4`, and `Hopper-v4`, where the expert data is also provided.

The performance of the expert policy can be found in `Initial_DataCollection_AverageReturn` in the log output.

Once you implement `TODO` above, you can train a BC policy for the Ant task as follows:

```
python aai4160/scripts/run_hw1.py \
    --expert_policy_file aai4160/policies/experts/Ant.pkl \
    --env_name Ant-v4 --exp_name bc_ant --n_iter 1 \
    --expert_data aai4160/expert_data/expert_data_Ant-v4.pkl \
    --video_log_freq -1
```

If your run succeeds, you will be able to find your tensorboard log data in `hw1_starter_code/data/q1_[--exp_name]_[--env_name]_[current_time]/`.

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. **Provide these details** (and any others you feel are appropriate) in the table caption.

Note: What "report the mean and standard deviation" means is that your `eval_batch_size` should be greater than `ep_len`, such 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 episodes (maybe more if any of them terminate early), and the logged `Eval_AverageReturn` and `Eval_StdReturn` represents the mean/std of your policy over these 5 rollouts. Make sure you include these parameters in the table caption as well.

Note: To generate videos of the policy rollouts, remove the flag "`--video_log_freq -1`". However, this is

slower, and so you probably want to keep this flag on while debugging.

4.1.1 BC Result

Fill in the values in the template table below.

Table 1: **Your caption goes here. Please include training details here.**

Environment	Performance (Mean Return $\pm$ Std)
`Ant-v4`	-1 $\pm$ -1
`[Env you've chosen]`	-1 $\pm$ -1

4.2 Hyperparameter Tuning of Behavioral Cloning

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 rationale for why you chose it.

You should include at least **4 different** settings for the hyperparameter you have chosen, including the default setting you used in the previous part.

Note: There are some default hyperparameters you can specify using the command line arguments. You may want to choose one of the hyperparameters listed below:

- Number of gradient steps for training policy (`--num_agent_train_steps_per_iter`, default: 1000)
- The amount of training data (`--batch_size`, default: 1000)
- Training batch size (`--train_batch_size`, default: 100)
- Depth of the policy neural net (`--n_layers`, default: 2)
- Width of the policy neural net (`--size`, default: 64)
- Learning rate for supervised learning (`--learning_rate`, default: $5e-3$)

You can specify the hyperparameter in the command line when you execute the script. For example, if you run the command like this, you can train the policy for 500 gradient steps:

```
python aai4160/scripts/run_hw1.py \
    --num_agent_train_steps_per_iter 500 \
    --some other arguments...,
```

Note: Use `matplotlib` for drawing the plots. If you are not familiar with `matplotlib`, you can refer to its official tutorial.

4.2.1 Hyperparameter Tuning Results

Hyperparameter: **Write the hyperparameter you have chosen and four different values you have tested.**

Plot:

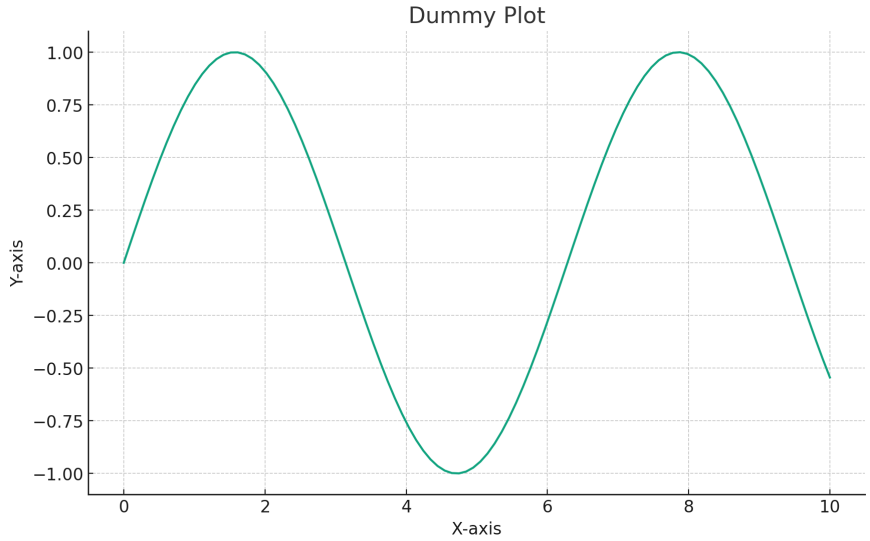

Figure 1: **Replace this with your plot, and add the caption. PDF is preferred, but PNG and JPG are also okay.**

Rationale: **Your rationale here.**

5 DAgger

5.1 Implement DAgger

Now your task is to implement the DAgger algorithm. If you implemented the BC part correctly, you can just implement the `TODO` in the following file.

- `do_relabel_with_expert` function in `aai4160/infrastructure/bc_trainer.py`

Once you have filled in all of the instructions specified with `TODO` comments in the code, you should be able to train DAgger with the following command:

```
python aai4160/scripts/run_hw1.py \
  --expert_policy_file aai4160/policies/experts/Ant.pkl \
  --env_name Ant-v4 --exp_name dagger_ant --n_iter 10 \
  --do_dagger \
  --expert_data aai4160/expert_data/expert_data_Ant-v4.pkl \
  --video_log_freq -1
```

5.2 Compare BC and DAgger

Run DAgger and report results on the two tasks you tested previously with BC (i.e., Ant + another environment). Report your results in the form of a learning curve, plotting the number of DAgger iterations vs. the policy's mean return. In the caption, state which task you used, and any details regarding network architecture, amount of data, etc. (as in the previous section).

Note: You can use the example helper script (`aai4160/scripts/parse_tensorboard.py`) to parse the data from the tensorboard logs and plot the figure. Here's an example usage that saves the figure as `output_plot.png`:

```
python aai4160/scripts/parse_tensorboard.py \
  --input_log_files data/[replace_here_with_the_name_of_log_folder] \
  --data_key "Eval_AverageReturn" \
  --title "DAgger: Ant-v4" \
  --x_label_name "DAgger iterations" \
  --y_label_name "Mean Return" \
  --output_file "output_plot.png"
```

You may also want to plot the performances of BC and expert policy as a horizontal line and plot the standard deviations as the error bars. Feel free to modify the example parsing script as you want.

5.2.1 DAgger Result

Include the plots of the two tasks here.

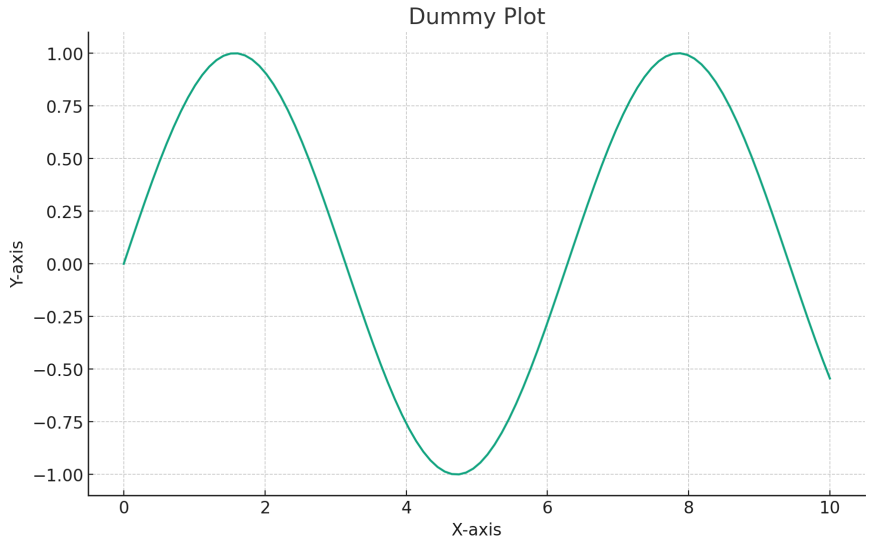

Figure 2: **Replace this with your plot, and add the caption.**

5.2.2 Justification of the Result

Compare the performances of DAgger, BC and expert policies, and explain the results.

6 Discussion

Please provide us a rough estimate, in hours, for each problem, how much time you spent. This will help us calibrate the difficulty for future homework.

- **Behavioral Cloning**: **XX hours**
- **DAgger**: **XX hours**

Feel free to share your feedback here, if any: **We would really appreciate your feedback to improve the reinforcement learning class.**

7 Submission

Please include the code, tensorboard logs data, and the report. Zip it to `hw1_[YourStudentID].zip`. The structure of the submission file should be:

```
hw1_YourStudentId.zip
├── hw1_YourStudentId.pdf
├── aai4160/
│   └── ...codes
└── data/
    └── ...tensorboard log folders
```

Note: Do NOT include the videos (.mp4 files) in your submission. Your submission file size should be less than 15MB.