CS 593 RL1 Fall 2025
Assignment 1: Behavior Cloning

Purdue University

Due September 24 2025, 11:59 PM

Installation

Setting up a Conda Environment

1. Open a terminal/command prompt and create a new environment:

```
conda create -n cs593rl python=3.9 -y
```

2. Activate the environment:

```
conda activate cs593rl
```

3. Navigate to the directory where you unzipped the code and install the necessary Python packages from `requirements.txt`:

```
pip install -r requirements.txt
```

This assigmnent uses two Gymnasium environments: Cartpole and Blackjack. Refer to the documentation pages linked above for information on the state space, action space, and reward function (though in this assignment rewards are only reported as part of the evaluation).

Part 1: Implement Behavior Cloning

As discussed in class, behavior cloning consists of two steps:

1. Collect demonstrations.
2. Use supervised learning to train a policy over the demonstrations.

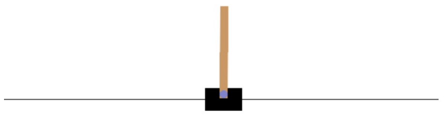

Figure 1: Cartpole

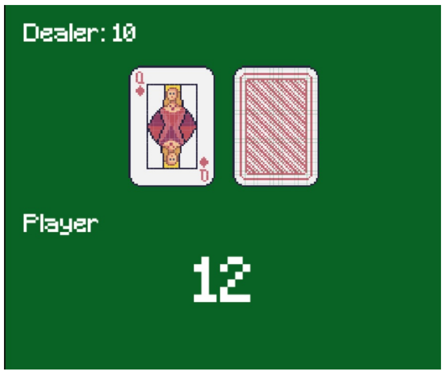

Figure 2: Blackjack

We have provided a starter implementation of behavior cloning, with some functionality unimplemented, along with example demonstrations. You have the option of running the code either on your own machine, on a cloud based service (such as Google Colab), or on RCAC Scholar.
Use the concepts introduced in class along with the Gymnasium/PyTorch documentation to fill in the blanks in the code marked with `TODO`. Commands for running behavior cloning are given in the README.md, within the corresponding source folder. We recommend that you read the files in the following order. You will need to fill in some components, labeled `TODO` in the code, and listed out here.

- `main.py`
  - Write code for loading demonstrations in `train_imitation()`
- `imitation.py`
  – Write code for initializing the policy network in `__init__()`.
  – Write code for performing optimization and computing metrics in `_train_epoch()`.
- `utils.py`
  - Write code for sampling actions and executing them in `evaluate_policy()`.

1. Behavior Cloning [5 pts]

Run behavior cloning using the following command and report results on both tasks: Cartpole and Blackjack. You should observe increasing training accuracy, decreasing training loss, and increasing evaluation reward. Here is how you can run each task:

```
python main.py --env CartPole-v1
python main.py --env Blackjack-v1
```

When prompted to pick a demonstration file, pick the demonstrations corresponding to your environment with `expert` in the file name.

Tip: You can plot performance metrics along with renders of policy rollouts by viewing the results in Tensorboard by running the following command and then navigating to `http://localhost:6006/` in a browser.
Report the training accuracy and evaluation reward plots for both tasks in the report.

```
    tensorboard --logdir runs
```

2. Hyperparameter sensitivity analysis [5 pts]

Experiment with two of the following hyperparameters and observe how the values affect the performance of the behavior cloning agent: learning rate, batch size, number of training epochs. For one of the tasks used in the previous question, show training accuracy and evaluation reward plots for each hyperparameter, with 5 different training runs in each plot where each run corresponds to a different hyperparameter value (you can obtain these from Tensorboard).

For example, if you choose to vary learning rate, you may choose values such as 1e-1, 1e-2, 1e-3, 1e-4, 1e-5 and both your accuracy/reward plots should have 5 runs corresponding to these values.

You can change these values by using the following commands:

```
python main.py --env CartPole-v1 --lr=1e-5
python main.py --env CartPole-v1 --batch_size=64
python main.py --env CartPole-v1 --num_epochs=50
```

Make sure to identify which hyperparameters you have changed, and which runs correspond to which values. You can either indicate this in the report or change the name of the log file in the `runs` directory, which then updates the label name in Tensorboard.

Tip: You can toggle certain runs as visible or invisible by clicking the checkbox next to run name on the left of the Tensorboard window.

Part 2: Collect and train over new data [5 pts]

We have provided the ability for you to collect your own demonstrations by playing each game using keyboard inputs. You may do so by running the following commands:

```
python main.py --env CartPole-v1 --collect
python main.py --env Blackjack-v1 --collect
```

To use the new demonstrations during training, make sure to select the demonstration files with `human` in the name when prompted. For this question, collect at least 20 demonstrations for one of the tasks (Cartpole is rather hard, so we recommend Blackjack) and train a policy. For fun see if you can train one that matches or exceeds the performance of the policy you

trained with the provided demonstrations. As this is skill based, we will not grade you on the quality of your demonstrations (and resulting policy) as long as you can show that the accuracy of the trained model improves, meaning it fits the observed data.
As in Question 1.1, report the training accuracy and evaluation reward plots for your new model.

Submission Instructions

Using Latex or another text editor, make a PDF report containing: 4 figures for Question 1.1 (corresponding to acc/reward for both tasks), 4 figures for Question 1.2, and 2 figures for Question 2. For simplicity, you may download/screenshot the Tensorboard figures and include them directly in the report.
In order to turn in your code and experiment logs, create a zip folder that contains the following:

- Your PDF report.
- Your completed code, including the data files you have collected.

Upload the resulting zip file to Brightspace.