Homework 1

CMU 10-703: Deep Reinforcement Learning (Fall 2019)
OUT: Sept. 6, 2019
DUE: Sept. 20, 2019 by 11:59pm

Instructions: START HERE

- **Collaboration policy:** You may work in groups of up to three people for this assignment. It is also OK to get clarification (but not solutions) from books or online resources after you have thought about the problems on your own. You are expected to comply with the University Policy on Academic Integrity and Plagiarism[1].
- **Late Submission Policy:** You are allowed a total of 10 grace days for your homeworks. However, no more than 3 grace days may be applied to a single assignment. Any assignment submitted after 3 days will not receive any credit. Grace days do not need to be requested or mentioned in emails; we will automatically apply them to students who submit late. We will not give any further extensions so make sure you only use them when you are absolutely sure you need them. See the Assignments and Grading Policy here for more information about grace days and late submissions: `https://cmudeeprl.github.io/703website/logistics/`
- **Submitting your work:**
  - **Gradescope:** Please write your answers and copy your plots into the provided LaTeX template, and upload a PDF to the GradeScope assignment titled "Homework 1." Additionally, export the code from your Colab notebook ([File → Export .py]) and upload it the GradeScope assignment titled "Homework 1: Code." Each team should only upload one copy of each part. Regrade requests can be made within one week of the assignment being graded.
  - **Autolab:** Autolab is not used for this assignment.

[1] `https://www.cmu.edu/policies/`

Problem 0: Collaborators

Please list your name and Andrew ID, as well as those of your collaborators.

Problem 1: Behavior Cloning and DAGGER (50 pt)

In this problem, you will implement behavior cloning using supervised imitation learning from an expert policy. This problem, and the rest of the assignment, will be implemented in the following Colab notebook:

`https://colab.research.google.com/drive/1pz2vH_FPkikitEtS5GXkzk7QzKm3ZwqH`

This Colab notebook runs in the cloud, so you should not need to install anything on your laptop.

Background: Imitation Module

We have provided you with some function templates in the Colab notebook that you should implement. If you need to modify the function signatures, you may do so, but specify in your report what you changed and why.

Preliminaries (0pt)

First, you will implement a TensorFlow/Keras model. You will use this model a number of times in the subsequent problems. To test that your model is implemented correctly, you will use it to solve a toy classification task.

Next, you will implement a function to collect data using a policy in an environment. This function will be used in subsequent problems.

Behavior Cloning (25 pt)

Start by implementing the `train()` method of the `Imitation` class. To test your behavior cloning implementation, you will run the following cell titled "Experiment: Student vs Expert."

1. [**15 pts**] Run your behavior cloning implementation for 100 iterations. Plot the reward, training loss, and training accuracy throughout training.

2. [**10pts**] This question studies how the amount of expert data effects the performance. You will run the same experiment as above, each time varying the number of expert episodes collected at each iteration. Use values of 1, 10, 50, and 100. As before, plot the reward, loss, and accuracy for each, remembering to label each line.

DAGGER (25 pt)

In the previous problem, you saw that when the cloned agent is in states far from normal expert demonstration states, it does a worse job of controlling the cart-pole than the expert. In this problem you will implement the DAGGER algorithm [4]. Implementing DAGGER is quite straightforward. First, implement the `generate_dagger_data()` method of the `Imitation` class. Second, in the cell titled "Experiment: Student vs Expert," set `mode = dagger`.

1. [**10 pts**] Run your DAGGER implementation for 100 iterations. Plot the reward, training loss, and training accuracy throughout training. We have included code for plotting in the following cell.
2. [**10pts**] This question studies how the amount of expert data effects performance. You will run the same experiment as above, each time varying the number of expert episodes collected at each iteration. Use values of 1, 10, 50, and 100. As before, plot the reward, loss, and accuracy for each, remembering to label each line.
3. [**5pts**] Does your DAGGER implementation outperform your behavior cloning implementation? Generate a hypothesis to explain this observation. What experiment could you run to test this hypothesis?

Problem 2: CMA-ES (25 pts)

For this problem, you will be working with the Cartpole (`Cartpole-v0`) environment from OpenAI gym. You should already have the environment set up and ready to go after working with gym in the first problem. Please take a look at the implementation to learn more about the state and action spaces: `https://github.com/openai/gym/blob/master/gym/envs/classic_control/cartpole.py`.

For Cartpole, we have continuous states and hence we cannot use policy learning methods for finite MDPs. In class, we learned about covariance matrix adaptation evolutionary strategy (CMA-ES), a black box, gradient-free optimization method. In this problem, you will implement CMA-ES find a policy for the Cartpole environment. There is template code provided for CMA-ES in the Colab notebook. Start by filling in the missing pieces in the `CMAES` class.

1. [**15 pts**] Run CMA-ES on Cartpole for 200 iterations (or until it reaches a reward of 200), using the provided hyperparameters. Plot the best point reward and average point reward throughout training.
2. [**10 pts**] Run your CMA-ES algorithm with population size of 20, 50, and 100. For each of the population sizes, run CMA-ES for 200 iterations (or until it reaches a reward of 200) and plot the best point reward at each iteration. In another plot, plot the best point reward as a function of the number of weights evaluated (# of iterations $\times$ population size).
3. (Optional) We provided most of the hyperparameters to you for this problem. However, in practice, significant time is spent tuning hyperparameters. Play around with the dimensionality of your network as well as other parameters of CMA-ES to get a feel for how the algorithm behaves.

Problem 3: GAIL (25 pts)

In this last part, you will implement Generative Adversarial Imitation Learning (GAIL) [2]. While the behavior cloning you did in the first part was based on supervised learning, GAIL is based on generative modeling. Recall that GAIL simultaneously learns a discriminator (using supervised learning) and a policy (using reinforcement learning). While the original GAIL paper used TRPO as its RL algorithm, we will use CMA-ES instead.

After completing this question, you should have a good grasp on the differences between behavior cloning and GAIL. As before, we have provided template code in the Colab notebook. To get started, implement the `collect_data()` and `train_discriminator()` methods of the `GAIL` class.

1. [**5pts**] The key component of GAIL is a discriminator that attempts to distinguish trajectories generated by the expert from trajectories generated by your policy. To start, try training just the discriminator (not the policy), and include a plot of the training accuracy. You should get an accuracy of at least 90%.

2. [**5pts**] Now, you will train a policy with CMA-ES, using the discriminator you trained in the previous problem as your reward function:

$$r(s, a) = \log p(\texttt{EXPERT} \mid s)$$

   We'll use two metrics to evaluate how "close" your policy is to the expert policy. The first metric is the cumulative task reward within a single episode. The second metric looks at the distribution over the X-coordinate of the cart. Discretize the X coordinate into 10 equal sized bins, and compute how frequently your policy and the expert policy visits each bin. Measure the discrepancy between these two distributions using Total Variation (TV) distance:

$$TV(p(x)\|q(x)) \triangleq \frac{1}{2}\sum_x |p(x) - q(x)|$$

   Run CMA-ES for 100 iterations, and plot the task reward (not the discriminator reward) and the TV distance throughout training.

3. [**5pts**] Now, you will implement the full GAIL algorithm, alternating between updating the discriminator and updating the policy. In our implementation, we found it useful to update the discriminator only once every 10 steps. As before, record the task reward and TV distance over 100 iterations of training, and include the plot in your submission.

4. [**5pts**] The frequency with which you update the discriminator is an important hyperparameter. Try varying the discriminator update rate. Describe what you find, including at least one plot to support your findings.

5. [**5pts**] In this homework, you have implemented three imitation learning algorithms: behavior cloning, DAGGER, and GAIL. Which did you find worked the best? On what sorts of tasks do you expect each method to work best?

Extra (2pt)

Feedback (1pt): You can help the course staff improve the course by providing feedback. You will receive a point if you provide actionable feedback. What was the most confusing part of this homework, and what would have made it less confusing?

Time Spent (1pt): How many hours did you spend working on this assignment? Your answer will not affect your grade.

References

[1] J Andrew Bagnell. An invitation to imitation. Technical report, DTIC Document, 2015.

[2] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pages 4565–4573, 2016.

[3] Stephane Ross. Interactive learning for sequential decisions and predictions. 2013.

[4] Stéphane Ross, Geoffrey J Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, volume 1, page 6, 2011.