6.884 Spring 2020 Computational Sensorimotor Learning HW 3

DUE DATE: 05/07/2020 (by 2.30pm ET)

Introduction

Overview

In reinforcement learning, an agent hopes to stumble upon rewards by taking a sequence of random actions and reinforces actions that lead to high rewards. However, the random exploration is data inefficient. In this HW, we will use the expert demonstration to improve the data efficiency.

Pushing Environment details

In this environment, we will use low-dimensional state representation composed of the location of the tip of the arm, the object location, and the goal location. The robot acts by changing its tip position in the x-y plane. The pushing environment is contained in `pusher-goal.py`.

Expert Dataset

We have provided you with an expert dataset. In practice, it's hard to give perfect demonstrations and to emulate this, we have provided you with sub-optimal demonstrations. The file `expert-data.py` has code for reading the expert dataset. Note that you are free to use a library of your choice and not necessarily pyTorch.

1 Problem 1: Learning a policy with Behavioral Cloning

In this problem, we will learn a policy for pushing using the `expert dataset`. We will use the learned policy for pushing an object.

Deliverables:

- Source code used for learning policy with behavioral cloning (20 pts).
- A detailed description of policy architecture and loss function used for training (5 pts).
- Training plot showing loss as a function of time. Evaluate the learned policy on 100 episodes and report the average L2 distance between the object location and the goal location. An average L2 distance of 0.22 (or less) should be good (15 pts).
- Video showing evaluation of policy on 10 episodes. (10 pts).

2 Problem 2: Fine-tuning with Reinforcement Learning

In this problem, we will fine-tune the policy learned using the expert dataset with PPO. You can use your implementation from HW1.

To ensure that we don't stray too far from the expert demonstrations, we can modify the PPO objective function to include $\lambda E_{(s,a)\sim\text{expert}}[\log \pi_\theta(a|s)]$ where π_θ is the policy being fine-tuned and λ is a hyper-parameter. Compare the joint loss fine-tuning to the vanilla fine-tuning where $\lambda = 0$.

Once you have fine-tuned the policy, you should learn another policy from scratch and compare its learning curve to that of the fine-tuned policy.

Deliverables:

- Source code used for fine-tuning the learned policy and learning a policy from scratch (20 pts).
- Learning curve of the vanilla fine-tuned policy, joint loss fine-tuned policy and the policy learned form scratch. Evaluate the three policies on 100 episodes and report the average L2 distance between the object location and the goal location (20 pts).
- Video showing evaluation of the three policies on 10 episodes (10 pts).

3 Submission Instructions

The deliverables mentioned above should be submitted on gradescope. Please also submit your code (along with the instructions on how to run it) in a zip file on gradescope.