6.884 Spring 2020 Computational Sensorimotor Learning HW 2

DUE DATE: 04/09/2020 (by 2.30pm ET)

Introduction

0.1 Overview

In HW 1, we used model-free reinforcement learning to train agents for reaching. Alternatively, agents can also be trained via self-supervision to learn a model of the world, and then plan with this learned model. Broadly, there are two types of models: forward and inverse model. The forward model predicts future state given current state and action. The inverse model predicts action given the current state and the desired future state. In this HW, we will implement learning of forward and inverse models.

0.2 Pushing Environment details

The action space is four-dimensional and consists of the initial and the final end-effector positions of the robot's arm tip. The state space consists of two-dimensional position of the object.

We have provided a pushing environment contained in `push-env.py`. You have to use the current object pose and the sampled goal object pose to perform a push with your learned inverse/forward model, and compute the distance between the final object pose after the push and the goal object pose. We have provided you with some starter code and you need to complete the function `plan-inverse-model` (in `push-env.py`) to sample different goals, and perform the push with your learned model.

0.3 Dataset

While typically learning requires data-collection, we have simplified this process and provided you with a `pushing dataset` consisting of 70k pushes for training and 70k pushes for testing. Another helper file `dataset.py`, creates a torch dataset object for reading the pushing dataset. Note that you are free to use a library of your choice and not necessarily pyTorch.

1 Problem 1: Learning Inverse Model

In this problem, we will learn an inverse model for pushing using the `pushing dataset`. We will use the learned inverse model for pushing an object.

Deliverables:

- Source code used for training inverse model (20 pts).
- A detailed description of inverse model architecture and loss function used for training (5 pts).

- Training plot showing loss as a function of time. Final train and test loss (5 pts).
- Video showing a ground truth push and the corresponding push performed by the inverse model. Report the distance between the final object pose after the push and the goal object pose. Sample 10 such pushes (20 pts).

2 Problem 2: Learning forward model

In this problem, we will learn a forward model for pushing using the `pushing dataset`. We will use the learned forward model to plan for pushing an object using Cross Entropy Method (CEM).

NOTE: For CEM algorithm, we recommend sampling push angle and push length and then using the push angle and push length to calculate the initial and the final end-effector positions of the robot's arm tip. You can look at `sample-push` function in `push-env.py`.

Deliverables:

- Source code used for training forward model and for planning using CEM (20 pts).
- A detailed description of forward model architecture and loss function used for training (5 pts).
- Training plot showing loss as a function of time. Final train and test loss (5 pts).
- Video showing a ground truth push and the corresponding push performed by the forward model. Report the distance between the final object pose after the push and the goal object pose. Sample 10 such pushes (20 pts).

3 Problem 3: Extrapolating with learned model

In the last two problems, we learned to model the primitive skill of pushing. Ideally, we would like to use the pushing skill to perform more complex tasks. However, the learned models might not succeed in complex tasks due to the change in the data distribution. In this problem, we will test the ability of the learned models to extrapolate to multi-step pushing. You need to complete the function `plan-inverse-model-extrapolate` (in `push-env.py`) to sample different goals, and perform the two-step push with your learned model.

Deliverables:

- Video showing a ground truth two-step push and the corresponding two-step push performed by the inverse model. Compute the distance between the final object pose after the push and the goal object pose. Sample 10 such pushes (20 pts).
- Video showing a ground truth two-step push and the corresponding two-step push performed by the forward model. Report the distance between the final object pose after the push and the goal object pose. Sample 10 such pushes (20 pts).
- Comment on how the forward and the inverse model extrapolate and how they compare (10 pts).

4 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.