Instruction for Project 2

Behavior Cloning for RC car racing

1. Introduction

In this assignment, you will acquire proficiency in the utilization of a specific behavior cloning algorithm called Gaussian Process Regression for the purpose of controlling an RC car. Gaussian Process Regression(GPR) necessitates the accumulation of expert demonstrations for training, and you should collect these demonstrations through the utilization of the controller developed in the previous project(Project 1). To augment the robustness of your model, it is advisable to generate a new map by referring to the provided 'map generation tutorial'. OpenAI Gym, renowned for its versatility in facilitating the development and comparative analysis of reinforcement learning algorithms, will be used to form our code. The overarching objective is the establishment of a fundamental behavior cloning algorithm for RC car racing.

2. Algorithm

A brief description of BC for a general configuration space is shown in Figure 2. We assume the environment as Markov Decision Process (MDP). Set of states S, set of actions A, and transition model $P(s, a)$ express the environment. τ indicates the agent's policy and $\tau^* = (s_0^*, a_0^*, s_1^*, a_1^*, \ldots)$ indicates the trajectories created by expert's demonstrations. With given τ^*, we treat the state-action pairs as iid (independent and identically distributed) and apply Gaussian Process Regression (GPR).

1. Collect demonstrations (τ^* trajectories) from expert
2. Treat the demonstrations as i.i.d. state-action pairs: $(s_0^*, a_0^*), (s_1^*, a_1^*), \ldots$
3. Learn π_θ policy using supervised learning by minimizing the loss function $L\big(a^*, \pi_\theta(s)\big)$

Figure 2. Pseudo code of BC

In the following paragraphs, we will explain each step of the BC algorithm.

Step 1. Initialize the gym environment.

Step 2. Load expert's demonstration observations and actions. Normalize the data by obtaining their mean and standard variance values. Additionally, shuffle data to break correlation between themselves. We recommend you to use scikit-learn packages for implementation of codes.

Step 3. Fit the GPR model using normalized and shuffled demonstration data as targets.

Step 4. For n steps, choose actions from the fitted GPR model and accumulate each step's reward.

Step 5. After the environment terminates (done = True), print the total reward.

3. Explanation of Code

RLLAB_project2.py

- GaussianProcess class
  - __init__ (self, args)
  - load (self)
  - get_action (self, obs)
  - callback_query(self, data)
- __main__()

GaussianProcess class defines the behavior cloning model and demo data's load function.

GaussianProcess.__init__

: Initialize the agent and GPR model. You can change the RBF kernel or other parameters to set the GPR model.

GaussianProcess.load

: Load the expert's demonstration data and fit into the defined GPR model. Data should be loaded as a dictionary structure with keys {'states', 'actions', 'rewards'}. Calculate the mean and standard variance of both demo observations and actions to normalize those values. To break the correlation among data sequences, shuffle the orders.

GaussianProcess.get_action

: Predict the agent's action for current observation by the fitted GPR model. The GPR model indicates the trained policy from **GaussianProcess.load** function. **Make sure to post-process the input observation's 1083-dim raw data appropriately to achieve high performance.**

GaussianProcess.callback_query

: Send a query to the ROS environment and publish evaluation results. After evaluation, performance measurements such as (passed waypoints)/(total waypoints) and lap time will be displayed on the web page's leaderboard.

__main__

: Initialize gym environment and train the BC model using expert's demonstration data. Until the environment terminates, calculate the reward for each step. If the experiment is done, print the total reward.

To do this project, you should follow the steps below.

1) Collect the expert demonstration(state, action) using the controller you made in Project 1.
2) **Change the TEAM_NAME on the top of the code to your team's name. Otherwise, we can't grade your code**
3) Implement __init__, load and get_action functions of GaussianProcess class. Follow the commented descriptions in the code.
4) Evaluate your model. You can collect more data in various maps to obtain higher performance of your model.
5) Good Luck~

4. Submission format

To evaluate your code in a local PC, follow the instructions described in the class repository(https://github.com/rllab-snu/Intelligent-Systems-2023).

Commit and push your codes on github and make a query on our project web server(http://147.46.116.112:36507/).