CMSC 818B: Decision-Making for Robotics (Fall 2019)

Homework 3

Due: November 7th, 11:59PM

October 24, 2019

You are allowed to work on this homework in groups of two students. Individual submissions are also okay. Your submission must be your (or your group's) original work. Each member of the group must submit on ELMS. Please follow the submission instructions posted on ELMS exactly.

Problem 1 — 100 points

Implement Q-Learning to solve the MountainCar-v0 problem from OpenAI gym. Information about the environment is provided online here: `https://github.com/openai/gym/wiki/MountainCar-v0`. The state space for this problem is the same as the observation space listed in the link above. That is, the state of the agent is given by its position and velocity. The agent can take three discrete actions.

You must use either Gaussian Process (GP) regression or a neural network (NN) as a function approximator. The hyperparameters (kernel for GP, architecture for NN) are your design choice.

In MountainCar-v0, you get a reward of -1 for every action until the end of the episode. The episode ends either when you reach the goal position or after 200 steps (whichever comes earlier). Therefore, an "unsuccessful" episode will give you a reward of -200. The sooner you reach the goal, the lower your reward will be. Therefore, we will use the running average of the reward of the most recent 100 episodes as one of our performance metrics.

To help you get started, a starter file is provided on ELMS. The file was tested on Python 3.17 using OpenAI gym version 0.14.0. It is likely compatible with other versions as well.

Submission Instructions.

- You must submit a pdf report that gives a description of your implementation. In particular:
  - Explain and justify[1] which kernel function you used in the case of GP or the architecture that you selected for your NN (20 points).
  - Explain and justify your exploration-exploitation strategy (15 points).
  - Explain and justify your choice of all other hyperparameters (15 points): in particular, discount factor, learning rate, loss function, kernel hyperparameters, termination condition.
- In the pdf report, report how well did the function approximator approximate the Q-values. You can do this by reporting, for example, the mean square error for training data. (5 points)
- Submit a screencast video (either as a link in the pdf report or in the zipped submission folder) of the policy being executed on the agent at the start of the learning, at the end of the learning, and in the middle. You can turn off exploration to show these policies. (10 points)

[1] Justify your design choices either by explaining the rationale behind them or by describing your design selection process (e.g., what other options did you consider? what factors did you consider in making your choices?). Simply stating what you used is not sufficient to get full points.

- Submit your code as a Python named `problem1_sol.py`. Other supporting files can be submitted but the main one should be named exactly as described here. Submit a README that mentions the dependencies on the libraries. You are allowed to use standard GP or NN libraries (tf, Keras, PyTorch,pandas). If you are not sure whether your library is allowed, check with the instructor. (10 points)
- In the pdf report, describe the performance of your algorithm. This must include a plot of the reward obtained per episode as well as the running average of the reward obtained in the last 100 episodes. Also report the episode number where the running average of the reward drops below -175, -150, and below -100 (if applicable). Also report the reward for the final policy that is learned by your algorithm. To report this value, you can execute the policy by turning off exploration (e.g., by setting $\epsilon = 0$) so that you are only exploiting. (25 points)

In general, you are discouraged from changing the reward function. However, if you need to, you can change the reward function. If you do, explain the new reward function and your rationale behind it. Also show how it helped improve the performance of the algorithm. This will change some of the requirements in the last bullet – make appropriate changes depending on your reward function.

Bonus Problem **30 points**

Same as above but solve either the MountainCarContinuous-v0[2] or the BipedalWalker-v2[3] environments (all the submission points will be scaled down).

[2] `https://github.com/openai/gym/wiki/MountainCarContinuous-v0`
[3] `https://github.com/openai/gym/wiki/BipedalWalker-v2`