Robot Autonomy - Homework 4

Professor: Oliver Kroemer

Due: April 25th 2021, 11:59pm

1 Introduction

This homework will focus on reinforcement learning for a block throwing robot. We will first look at a script for running the cross entropy method. We will then complete two scripts: one for REPS and another for Bayesian optimization. The block throwing is performed in the VREP simulator. Open up the ThrowScene file in the simulation environment. We will be using a remote interface similar to the one used for the simulated locobot experiments. The robot_interface python file also includes the functions for rolling out a policy on the simulated robot and for computing the reward based on the outcome.

2 Cross Entropy Method (CEM)

We will begin by looking at the script CEM.py for running the cross entropy method. Note that this section does not involve altering the script, and you will just be running the python script. However, it is worth looking through the script to get an understanding of the algorithm. The REPS script will have the same structure as the CEM script (sample+evaluate theta, compute weights based on rewards, update policy according to weights, repeat). The Gaussian Process UCB script will use the CEM to optimize the acquisition function for the Bayesian optimization.

The script should be directly executable, and you should see the throwing performance gradually improve over time, generating a plot after a given number of policy updates.

3 Relative Entropy Policy Search (REPS)

The REPS algorithm is given in REPS.py. As noted before, the structure is the same as for CEM. The computeETA function is meant to optimize the dual function to determine the η "temperature" parameter. The optimization has largely been set up with scipy's optimize.minimize function. However, to perform the optimization, the dual function needs to provided. Insert the dual

function equation at the TODO. Note that the function takes in eta (the parameter to be optimized), but it also has access to the returns R. Also note that, for numerical stability, we have subtracted the max return from all of the returns, and we should therefore include a +np.max(returns) to the dual function. The optimization will then compute the η value based on the distribution of observed returns from the previous batch of rollouts.

Once the η parameter has been computed, the next step is to compute the weights of each of the samples based on their corresponding returns. These weights will subsequently be used for computing the mean and covariance matrix for the new updated policy. The weights are computed at the third TODO in the script. Implement the sample weighting employed by REPS.

To run the script, you need to complete one final TODO in the middle. This is the part of the script that will be evaluating the theta parameters by runinng them on the simulated robot. The function for executing the simulated task is given in the robot interface python file. It is an easy TODO to complete, and the goal is just makes sure that you follow the overall flow of the script.

The script should now be executable. You should see the robot becoming gradually better at throwing, and the script should ultimately create a plot of the rewards over time, which you should submit.

4 Bayesian Optimization (BO)

The last method that we will be looking at is Bayesian optimization using an upper confidence bound (UCB) acquisition function. This method will be implemented in GaussProcUCB.py. The script begins with a class implementing Gaussian processes. The amplitude, noise, and length scale hyperparameters are given to the GP at initiailization. We do not optimize the parameters (although ideally one would). The first TODO is in the training function. Given the training samples, compute the K matrix (aka, the Gram matrix) that captures the correlations between all of the training samples. We will be using a squared exponential kernel to compute the covariances based on the x_train inputs. The kernel has an amplitude of sigma_a**2 and a matrix of length scales given by np.diag(self.sigma_l)**2).

The next TODO is in the test function of the GP class. You need to compute the mean and standard deviation of the predicted Gaussian for the test input x. Use the inv_K matrix computed at training time to avoid repeated inversions. These two lines are the equations at the core of the Gaussian process computation.

The main part of the script begins by collecting two data samples to get started. The robot then uses CEM to optimize the acquisition function for determining the next sample. As the CEM is querying the GP rather than the robot, we use a lot more samples for this process. However, the robot needs to be told which acquisition function to optimize for to select the next action. Update the TODO line to use an Upper Confidence Bound (UCB) acquisition function with $\beta = 1$ such that the exploration and exploitation terms have equal

weights.

Once the CEM has completed, it should have focused in on a set of theta parameter values with a high acquisition value. The next step is to evaluate the selected parameters on the robot and compute the reward. Add the call to run the simulated rollout to the next TODO.

The final two TODO are simply to append the new data from running the rollout to the training data for recomputing the Gaussian process. The next iteration will then use the updated GP to compute the acquisition function.

The script should now be executable. You should see the robot becoming gradually better at throwing, and the script should ultimately create a plot of the rewards over time, which you should submit.

Submission:

For this homework, submit a pdf page with the following items:

- The reward plot from running the CEM script
- The reward plot from running the completed REPS script
- The reward plot from running the completed GP Bayesian Optimization script

As well as answers to the following questions:

1. What trend do you see in the CEM reward plot? Why do you think this is?
2. Did REPS have improved performance over CEM? Why or why not?
3. Did the Bayesian Optimization approach show an improvement over REPS?
4. Did the Bayesian Optimization approach perform as you expected? Why or why not?
5. What is one advantage and one disadvantage of each of the three methods?