Advanced Robotics: Homework 2

Deadline: November 20, 2024, 11:59 PM

Let us revisit the same navigation problem that you solved in HW 1, and consider the following MDP:

- Action $a_t = (f_{x,t}, f_{y,t})$, where $f_{x,t}$ is the force applied on the robot in the x-dimension, and $f_{y,t}$ is the force applied on the robot in the y-dimension. Both $f_{x,t}$ and $f_{y,t}$ are limited to values in the interval $[-1 \text{ Newton}, 1 \text{ Newton}]$.
- State $s_t = (x_t, y_t, \dot{x}_t, \dot{y}_t)$ is the position and velocity of the robot. At the beginning of each episode, the initial state is obtained by sampling the position of the robot uniformly in the workspace of the robot, and setting its initial velocity to 0.
- The transition function is defined as follows:

$$\begin{aligned} \dot{x}_{t+1} &= \dot{x}_t + (f_{x,t} - \rho_{x,t})\Delta t \\ \dot{y}_{t+1} &= \dot{y}_t + (f_{y,t} - \rho_{y,t})\Delta t \\ x_{t+1} &= x_t + \dot{x}_t \Delta t \\ y_{t+1} &= x_t + \dot{y}_t \Delta t \end{aligned}$$

  where $\rho_{x,t}$ and $\rho_{y,t}$ are small independent noises, sampled from $\mathcal{N}(0, 0.1)$ at each time-step t, and Δt is set to 0.1 seconds. We are assuming here that the robot has a mass of 1 Kg. You can imagine the force noises as air resistance or random friction (although air resistance should scale up as a function of velocity).
- The reward function is defined as: $R(s_t) = 1$ if $\|s_t - s_g\|_2 \leq \epsilon$, and $R(s_t) = 0$ otherwise. $s_g = (x_g, y_g, 0, 0)$. It's up to you to choose the values of the goal coordinates (x_g, y_g), but they should be fixed in advance. It's also up to you to choose a reasonable value for the goal threshold ϵ.
- Set the discount factor $\gamma = 0.99$.

What you need to do:

1. Leverage the Mujoco setup you had in HW 1 and modify it to simulate the MDP described above.

2. Design a small actor neural network that predicts actions $a_t \sim \pi_\theta(s_t)$ as Gaussians, and a second small critic network that predicts values $v_w(s_t)$ of policy π_θ.

3. Implement the actor-critic algorithm explained in slide 40 of the lecture "Policy Gradients and Actor Critics". It is up to you to define the length of the episodes (horizon) and the gradient step-sizes.

4. Report the average reward per step as a function of the number of episodes that you used for training. This is called the learning curve.

5. Submit the code and a small writeup on canvas. The writeup includes an explanation of what you did, and the results (the learning curve).