Project

A Robot That Teaches Itself To Play Basketball

Deadline: Monday, December 11, 23:59pm.
Perfect score: 100.

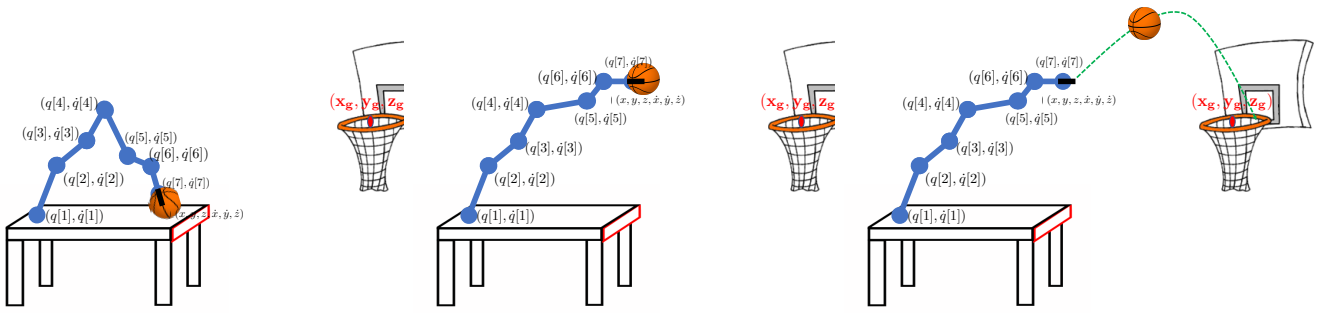
Assignment Instructions:

Submission Rules: Submit a compressed file that contains your code through Sakai (`sakai.rutgers.edu`).

Late Submissions: No late submission is allowed. 0 points for late assignments.

Collusion, Plagiarism, etc.: Each team must prepare its solutions independently from other teams, i.e., without using common notes, code or worksheets with other students or trying to solve problems in collaboration with other teams. You must indicate any external sources you have used in the preparation of your solution. Do not plagiarize online sources and in general make sure you do not violate any of the academic standards of the course, the department or the university.

Problem overview: Consider the seven-degrees-of-freedom robotic arm that you simulated in the previous assignment, illustrated in Figure 1. Let $(q_t[i], \dot{q}_t[i])$ refer to the angle and the angular velocity of joint i at time-step t, for $i \in \{1, 2, 3, 4, 5, 6, 7\}$ and $t \in \{1, \ldots, H\}$ where H is the maximum number of time-steps to run before stopping an episode and reseting the robotic arm to an initial configuration. The task of this robot is to throw a ball inside a hoop, mounted on a pole near the robot. Given the difficulty of programming the robot to perform this task by using the optimal control techniques we studied in class, and the difficulty of coming up with accurate models, we will follow a minimum effort approach and let the robot program itself to perform this task!



(a) Initial state of the robot (b) State of the robot at a given time-step t (c) State of the robot after releasing the ball

Figure 1: Starting from an initial configuration, described by the angles and the angular velocities of the robot's seven joints, find a policy that moves the robot to a configuration where the ball in hand is released. After releasing the ball, a reward is computed by simulating the trajectory of the ball and taking the distance of the ball to the center of the hoop. To achieve a maximum reward, a policy that returns joint velocities in each given configuration is learned from trials.

Steps:

1. Formalizing the task as a Markov Decision Process with continuous states and actions: the state space corresponds to the joint angles and velocities. An action is defined by an 8-dimensional vector, where the first seven elements define the change in velocity $\delta\dot{q}_t[i]$ that you want to apply on joint i, and the last element corresponds to a binary variable *release* where *release* = *true* means that the ball will be released in the next time-step and *release* = *false* means that the ball will still be held by the end-effector.

2. Once the ball is released, the robot receives an immediate reward, the trial is then ended, and the robot moves back to the initial configuration. The reward is computed by simulating the trajectory of the ball, and computing the distance

between the ball when it intersects the hoop's plan and hoop's center $(\mathbf{x_g}, \mathbf{y_g}, \mathbf{z_g})$. This is achieved by first computing the cartesian position and velocity of the end effector $(x, y, z, \dot{x}, \dot{y}, \dot{z})$ using the forward kinematic equations that we studied in the class, and using them as initial position and velocity in the parabolic equation of the projectile's motion.

3. Algorithm 1: Neural Fitted Q-Iteration with discretized actions. Refer to [Rie05] for an explanation of this method. Consider a discretized set of actions, for instance $\delta\dot{q}_t[i] \in \{-5°/s, 5°/s\}$. Neural Fitted Q-Iteration consists in using Q-learning while approximating the Q-value functions with a neural net and following an ϵ-greedy policy. Implement this algorithm and report two learning curves: (1) average reward per trial, (2) average score per trial. A score of 1 is given whenever the ball enters the hoop. The average score per trial is smaller than or equal to one. These curves are obtained by repeating all the experiments at least 5 times (from scratch) and taking the averages.

4. Algorithm 2: Policy Learning by Weighting Exploration with the Returns (PoWER). Refer to [KP11] for an explanation of this method. Implement this algorithm and report the two learning curves (as explained above).

5. Algorithm 3: If neither Algorithm 1 nor Algorithm 2 converges to an average score of 1, then you need to implement a third RL algorithm of your choice. You do not need to implement a third algorithm if you obtain good results with Algorithm 1 or with Algorithm 2.

6. Optional: add a small Gaussian noise to the state transitions and the ball's trajectory, and repeat the experiments with this new stochastic model.

The best performing team will be given the option to implement their algorithm on a real Kuka IIWA robot, with the assistance of the instructor and other grad students in the lab.

References

[KP11] Jens Kober and Jan Peters. Policy Search for Motor Primitives in Robotics. *Mach. Learn.*, 84(1-2):171–203, July 2011.

[Rie05] Martin Riedmiller. Neural fitted Q iteration: first experiences with a data efficient neural reinforcement learning method. In *In 16th European Conference on Machine Learning*, pages 317–328. Springer, 2005.

Have fun!