COMP0037 2022 / 2023 Robotic Systems
Lab 04: General Policy Iteration and Monte Carlo Policy Prediction

COMP0037 Teaching Team

February 23, 2024

Overview

In this lab, we will illustrate the material in Lectures 08 and 09 on GPI and Monte Carlo algorithms for policy prediction.

The scenario is similiar to that in Lab 03: some designers have learned their lesson and bought a robot from the Slightly Less McCheap Robot Company. The robot operates in an environment with goals and holes. Both are terminal sites — if the robot falls down them, it's game over. Goals give a big positive reward, holes a big negative one.

The robot is only capable of moving in four directions (`MOVE_LEFT`, `MOVE_RIGHT`, `MOVE_UP`, `MOVE_DOWN`) and waiting (`WAIT`). These first five actions are move actions. In addition, to these actions, we add two artificial ones. The `TERMINATE` action is only called at a terminal state (goal or hole) and `NONE` which is the only task allowed if the robot is in a state it shouldn't be able to get (e.g., inside a wall).

Installation Instructions

The code does not require you to install any additional packages beyond those required to support Lab 03.

Activities

1. The script `evaluate_gpi.py` is used to create a simple map and a simple policy and gives you a chance to see that GPI works correctly.

   The initial configuration is to run the policy evaluation algorithm, iterating over all the states.

   a. Run the script `evaluate_using_gpi.py` and make a note of the value of the value function which is generated.

   b. Modify the class `PolicyEvaluator` in `generalized_policy_iteration` to use a GPI iteration strategy such as modifying batches of random cells at a time. Recall that, for a globally optimal solution, whatever strategy you use has to (in the limit) sample all the cells an infinite number of times.

      Hint: You will need to revisit the conditions for determining if the solution has converged.

2. The script `evaluate_on_policy.py` uses an implementation of the on policy MC prediction algorithm with exploring starts to compute the state value function.

   a. Modify the function `_update_value_function_from_episode` to update the value function from each episode using the *any visit* strategy.

   b. Modify the function `_update_value_function_from_episode` to update the value function from each episode using the *first visit* strategy. Do you see much difference?

      Hint: I didn't...

   c. Try expanding the scenario to a larger size and check the behaviour of the algorithm. What do you see?

3. This question is about using off policy MC evalution. This requires two things: a behaviour policy, and the target policy.

   a. Run the script `visualize_random_policy.py` to see 50 samples of the random policy being sampled. What do you think is going on? Adjust the value of ϵ using the `set_epsilon` method to check the results.

   The script `evaluate_off_policy.py` uses an implementation of the off policy MC prediction algorithm with exploring starts to compute the state value function.

   b. Complete the implementation of `_update_value_function_from_episode` in the class `OffPolicyMCPredictor`.

   c. Compare the performance of the on and off policy predictors. What behaviour do you see?