

ASEN 6519-007 Decision Making under Uncertainty

Homework 6: Advanced POMDPs

March 22, 2021

1 Exercises

Question 1. (35 pts)

Write the following three elements of a POMDP solver:

1. A belief **Updater** and the associated **update** function to calculate the Bayesian belief update¹.
2. A **Policy struct** and associated **action** function that chooses an action based on a list of alpha vectors.
3. A function that calculates the QMDP alpha vectors for a POMDP and returns them as an object of the **Policy** type described above.

The starter code contains the skeleton code for these three items. Use your QMDP code to solve the **TigerPOMDP** from the **POMDPModels.jl** package and report the results along with those from the solver in the **SARSOP.jl** package.

Question 2. (25 pts)

Evaluate the following three policies on the cancer problem that you created in Homework 5:

1. The QMDP policy obtained using your code from Question 1 above.
2. A heuristic policy that outperforms the QMDP policy²
3. A near-optimal policy calculated by the SARSOP solver from the **SARSOP.jl** package.

First, report the results of these simulations in a table, then write a short paragraph answering the following question: Both the tiger and cancer problems are similar in that they have information-gathering actions. However, the gap between the QMDP and optimal solutions in one of the problems is much larger. Which problem has the larger gap, and why?

2 Challenge Problem

Question 3. (Lasertag POMDP, 40 pts)

In this problem, you will find a policy and belief updater for the laser tag POMDP model `HW6.LaserTagPOMDP()`. In this POMDP, a robot seeks to tag a moving target in a grid world with obstacles. The state space consists of all positions the robot and the target can take, and the problem ends with a reward of 100 as soon as they occupy the same cell. There are five actions, `:up`, `:down`, `:left`, `:right`, and `:measure`. The `:measure` action has a cost of -2 and gives the robot returns from lasers pointed in the four directions indicating the *exact* distance to the first wall, obstacle, or target that the laser encounters. When any other action is taken, the reward is -1 and the laser return for each direction is noisy; it is uniformly distributed between 0 and the

¹This can be a discrete Bayesian filter or a particle filter

²Hint: you may want to use the QMDP policy within your heuristic policy, i.e. take the QMDP actions some of the time.

distance to the nearest object. The `POMDPs.jl` interface including `POMDPModelTools.render` can be used to further explore the problem.

Submit a `Policy` or a tuple containing a `Policy` and a `Updater` to the `HW6.evaluate` function. A score of 35 will get full credit. You can use or modify your QMDP code from Question 1, or you are encouraged to use any tools available to you - any `POMDPs.jl` solvers, deep reinforcement learning, a particle filter from `ParticleFilters.jl`, a modification of your MCTS code from earlier homework, or heuristic policies are all acceptable. The `Policy` object may be a solution that was calculated offline, or an online planner.