ASEN 5264 Decision Making under Uncertainty
Homework 2: Markov Decision Processes

January 27, 2023

1 Conceptual Questions

Question 1. (10 pts)

a) Describe the difference between the reward function and the state-action (Q) value function.

b) Write down an equation for the state-action value in terms of R, T, and V.

Question 2. (20 pts) Consider a game with 3 squares in a horizontal line drawn on paper, a token, and a die. Each turn, the player can either reset or roll the die. If the player rolls and the die shows an odd number, the token is moved one square to the right, and if an even number is rolled, the token is moved two squares to the right (in both cases stopping at the rightmost square[1]). If the player resets, the token is always moved to the leftmost square. If the reset occurs when the token is in the middle square, two points are added; if the player resets when the token is on the right square, a point is subtracted.

a) Formulate this problem as an MDP (write down S, A, T, and R).

b) Calculate the optimal value when the token is in the leftmost square assuming a discount factor of $\gamma = 0.95$.[2]

c) Suppose you are not sure that the die is fair (i.e. whether it will yield odd and even with equal probability). Give finite upper and lower bounds for the accumulated discounted score that you can expect to receive with discount $\gamma = 0.95$.

2 Exercise

Question 3. (Value iteration for Grid World, 35 pts)

Solve the MDP `HW2.grid_world` with your own implementation of value iteration with a discount of $\gamma = 0.95$ and plot the resulting value function. All of the necessary information to solve this problem can be extracted with the `HW2.transition_matrices` and `HW2.reward_vectors` functions, and plotting can be accomplished with `POMDPModelTools.render(HW2.grid_world, color=v)` where `v` is the value function. See the starter code and function docstrings for more information.

3 Challenge Problem

Question 4. (Value iteration for ACAS, 35 pts)

[1] If the die is rolled from the middle square or right square, it will always end up in the right square.

[2] Hint: Guess an optimal policy, evaluate that optimal policy, and then verify that it satisfies the Bellman equation.

Your task is to find the optimal value function for an Aircraft Collision Avoidance System (ACAS). The encounter model will be specified as a Markov decision process, and your task will be to compute the value function for discount $\gamma = 0.99$ using value iteration or another suitable algorithm that you implement. The continuous physical state space will be discretized at various levels of granularity and the goal is to find the value function for the finest discretization possible.

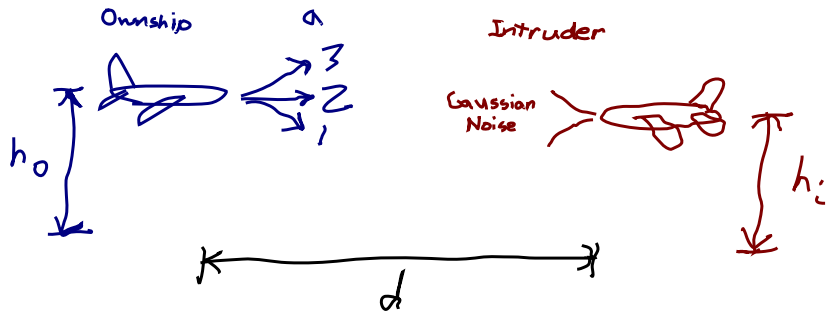
A model with discretization level `n` can be constructed with

```
m = HW2.UnresponsiveACASMDP(n)
```

The higher `n` is, the finer the discretization and the larger the state space. Again, all of the information needed to solve this problem can be extracted with the `HW2.transition_matrices` and `HW2.reward_vectors` functions, so you can start with your code from Question 3.

The score received for solving the problem is `n`. You must submit your code for this problem along with the `results.json` file from executing `HW2.evaluate(v, "email@colorado.edu")` where `v` is the value function vector[3]. A score of `n` = 7 or higher will receive full credit[4].

The information above this line is sufficient to complete this homework and receive full credit. However, the description of the model below may help to get the highest score on the leaderboard. The `UnresponsiveACASMDP` model implements the POMDPs.jl explicit MDP interface[5], and students are welcome to explore the problem further using that interface as well as ask on Ed and look at the source code for details. The underlying continuous model is defined as follows:



- The state space is 4-dimensional $\mathcal{S} = \mathbb{R}^4$, with each state consisting of $s = (h_o, \dot{h}_o, h_i, d)$ where h_o is the ownship altitude in feet, $\dot{h}_o$ is the rate of climb in ft/min, h_i is the intruder altitude in feet, and d is the distance between the aircraft in feet.
- The action space is $\mathcal{A} = \{-1500, 0, 1500\}$ and represents the change in rate of climb. The possible rates of climb are $\dot{h}_o \in \{-3000, -1500, 0, 1500, 3000\}$
- A reward of -100 is received for a near-mid-air collision, defined as the aircraft passing within 500 vertical feet and 100 horizontal feet of each other. Any change in rate of climb yields a reward of -1.
- The rate of climb, $\dot{h}_o$ changes instantly when an action is applied. Then the following dynamics are used: $d' = d - 2\,v\,\Delta t$ where v is the fixed horizontal velocity, $h'_o = h_o + \dot{h}_o\,\Delta t$, and $h'_i = h_i + W_{\Delta t \sigma^2}$ where W is the Wiener process[6]. Δt changes based on the discretization `n`.

[3] `HW2.evaluate` will check the value function with a tolerance of 1×10^{-6}

[4] By taking advantage of the structure of the problem, it is possible to attain a score of `n` = 20 with less than 10 minutes of computation time on a single core of a i7 laptop processor.

[5] The `states`, `actions`, `reward`, and `transition` functions from POMDPs.jl.

[6] `https://en.wikipedia.org/wiki/Wiener_process`