ASEN 5519-003 Decision Making under Uncertainty
Homework 3: Online MDP Methods

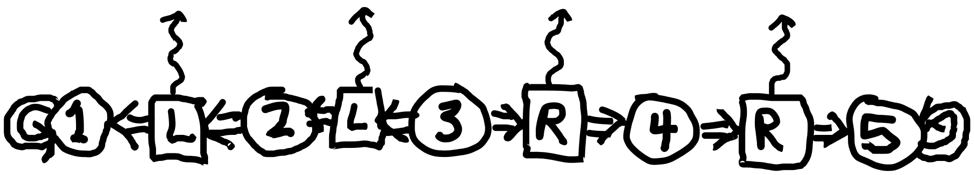
February 4, 2022

1 Conceptual Questions

Question 1. (20 pts) Do similar Q values imply similar rewards?

In the proof for Lemma 5 of the Sparse Sampling paper by Kearns, Mansour, and Ng,[1], the authors make the following claim: **if a policy π satisfies $|Q^*(s, \pi^*(s)) - Q^*(s, \pi(s))| \leq \beta$ for all $s \in \mathcal{S}$, then it immediately follows that $|\mathcal{R}(s, \pi^*(s)) - \mathcal{R}(s, \pi(s))| \leq \beta$.** In this exercise, you will demonstrate that this claim is mistaken.

Consider the MDP below:



The state space is $S = \{1, \ldots, 5\}$ and the action space is $A = \{L, R\}$ (but not all actions are available from each state). Transitions are deterministic as shown. The discount factor is $\gamma = 0.9$.

Choose a reward function, $\mathcal{R}$, (i.e. values for the squiggly arrows), a policy, π, and a value β that constitute a counterexample to the claim above. Justify your answer.

2 Exercises

Question 2. (30 pts) Monte Carlo Tree Search

Write code that performs 7 iterations of Monte Carlo Tree Search on an MDP created with `HW3.DenseGridWorld(seed=2)`, starting at state $(19, 19)$. You will need to produce three dictionaries:

- `Q` maps (s, a) tuples to Q value estimates.
- `N` maps (s, a) tuples to N, the number of times the node has been tried.
- `t` maps (s, a, s') tuples to the number of times that transition was generated during construction of the tree.

Then visualize the resulting tree with `HW3.visualize_tree(Q, N, t, SA[19, 19])`[2]. **Submit an image of the tree, the code used to generate it, and a few sentences describing the tree after 7 iterations** (e.g. which actions have the highest Q values? Does this make sense?).

You will need to use the following functions from POMDPs.jl for the problem:

[1] `https://www.cis.upenn.edu/~mkearns/papers/sparsesampling-journal.pdf`; Note: you do not need to read the paper to complete the problem.

[2] `SA` is from the `StaticArrays.jl` package.

- `actions(m)`
- `@gen(:sp, :r)(m, s, a)`
- `isterminal(m, s)`
- `discount(m)`
- `statetype(m)`
- `actiontype(m)`

(You may also wish to use `POMDPs.simulate` and `POMDPSimulators.RolloutSimulator` for the rollouts.)

`HW3.DenseGridWorld(seed=2)` generates a 60x60 grid world problem. There is a reward of +100 every 20 cells, i.e. at [20,20], [20,40], [40,20], etc. Once the agent reaches one of these reward cells, the problem terminates. All cells also have a cost generated by the seed.

Question 3. (20 pts) Planning with MCTS

In this exercise, you will compare MCTS with a heuristic policy and an optimal policy calculated using value iteration on the `DenseGridWorld(seed=3)` MDP.

a) Write a simulation loop for the MDP equivalent to the following:

```
r_total = 0.0
disc = 1.0
while !isterminal(mdp, s)
    a = :left # replace this with the heuristic policy or mcts!
    s, r = @gen(:sp,:r)(mdp, s, a)
    r_total += d*r
    d *= discount(mdp)
end
```

Use this code to evaluate a heuristic policy of your choice with 100 Monte Carlo simulations starting from state $(30, 30)$. Report the mean accumulated reward and standard error of the mean.

b) Use your Monte Carlo tree search from Question 2 to plan online in the simulation loop. Limit the planning time to 50ms. Evaluate the MCTS planner with 100 Monte Carlo simulations starting from state $(30, 30)$. Report the mean accumulated reward and standard error of the mean, along with a typical number of iterations that MCTS was able to complete each time it chose an action[3].

3 Challenge Problem

Question 4. (10 pts code and description, 20 pts score) Fast Online Planning

Create a function `select_action(m,s)` that takes in a 100×100 `DenseGridWorld`, `m`, and a state `s`, and returns a near-optimal action within 50ms. You may wish to base this code on the MCTS code that you wrote for Question 2. Evaluate this function with `HW3.evaluate` and **submit the resulting `json` file along with the code and a one paragraph to one page description of your approach**, including tuning parameters that worked well, the rollout policy, etc. A score of 50 will receive full credit. There are no restrictions on this problem - you may wish to use a different algorithm, multithreading, etc. Starter code will be linked to on Canvas that will give suggestions for timing and other details.

[3] You can determine a typical number of iterations by just printing out the number; you don't need to keep careful statistics unless you want to