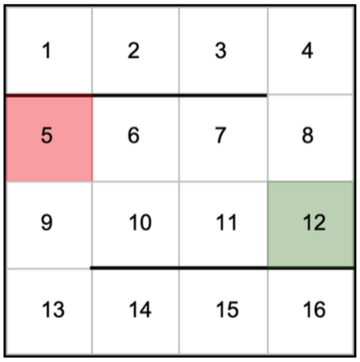
CSCI 6961: Machine Learning and Optimization

Problem Set on Reinforcement Learning

December 1, 2021

1 Gridworld [20 pts]

Consider the following grid environment. Starting from any unshaded square, you can move up, down, left, or right. Actions are deterministic and always succeed (e.g. going left from state 16 goes to state 15) unless they will cause the agent to run into a wall. The thicker edges indicate walls, and attempting to move in the direction of a wall results in staying in the same square (e.g. going in any direction other than left from state 16 stays in 16). Taking any action from the green target square (no. 12) earns a reward of r_g (so r(12, a) = r_g $\forall$ a) and ends the episode. Taking any action from the red square of death (no. 5) earns a reward of r_r (so r(5, a) = r_r $\forall$ a) and ends the episode. Otherwise, from every other square, taking any action is associated with a reward $r_s \in$ {-1, 0, +1} (even if the action results in the agent staying in the same square). Assume the discount factor γ = 1, r_g = +5, and r_r = -5 unless otherwise specified. [1]

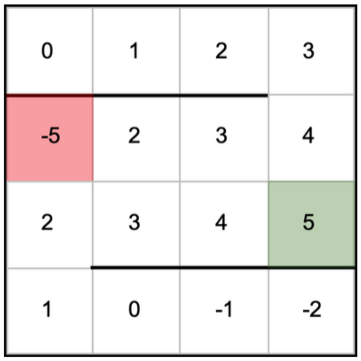


1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

(a) (4pts) Define the value of r_s that would cause the optimal policy to return the shortest path to the green target square (no. 12). Using this r_s, find the optimal value for each square.

Solution: By using $r_s = -1$ we get the following values in each square,

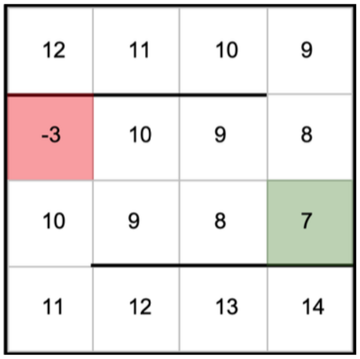
[1] Adopted from Stanford's Reinforcement Learning course (Winter 2020)



0	1	2	3
-5	2	3	4
2	3	4	5
1	0	-1	-2

(b) (4pts) Lets refer to the value function derived in (a) as $V_{old}^{\pi_g}$ and the policy as π_g. Suppose we are now in a new gridworld where all the rewards (r_s, r_g, and r_r) have +2 added to them. Consider still following π_g of the original gridworld, what will the values $V_{new}^{\pi_g}$ be in this second gridworld?

Solution:



12	11	10	9
-3	10	9	8
10	9	8	7
11	12	13	14

(c) (4pts) Consider a general MDP with rewards, and transitions. Consider a discount factor of γ. For this case assume that the horizon is infinite (so there is no termination). A policy π in this MDP induces a value function V^π (lets refer to this as V_{old}^{π}). Now suppose we have a new MDP where the only difference is that all rewards have a constant c added to them. Can you come up with an expression for the new value function V_{new}^{π} induced by π in this second MDP in terms of V_{old}^{π}, c, and γ?

Solution:

$$\begin{aligned}
V_{old}^{\pi}(s) &= E_\pi[\sum_{T=0}^{\infty} \gamma^T r_{t+T} | s_t = s] \\
V_{new}^{\pi}(s) &= E_\pi[\sum_{T=0}^{\infty} \gamma^T (r_{t+T} + c) | s_t = s] \\
&= E_\pi[\sum_{T=0}^{\infty} \gamma^T r_{t+T} | s_t = s] + c \sum_{T=0}^{\infty} \gamma^T \\
&= V_{old}^{\pi}(s) + \frac{c}{1-\gamma}
\end{aligned}$$

(d) (4pts) Lets go back to our gridworld from (a) with the default values for r_g, r_r, γ and with the value you specified for r_s. Suppose we now derived a second gridworld by adding a constant c to all rewards (r_s, r_g, and r_r) such that r_s = +2. How does the optimal policy change (Just give a one or two sentence description)? What do the values of the unshaded squares become?

Solution:

The optimal policy becomes a policy that would just wander around forever, never reaching either target. The value of all unshaded squares (and the green target square) become $+\infty$. The value of the red square of death is -2.

(e) (4pts) Lets take the second gridworld from part (d) and change γ such that $\gamma < 1$. What happens to the optimal policy now?

Solution:

For γ close to one the optimal policy is still to wander around forever. However as gamma decreases, there will be a point where the optimal policy switches to reach the green target square in the shortest time.

2 Programming [80 pts]

Now you will implement value iteration and policy iteration for the Gridworld problem. We have provided starter code for both along with the environment. [2]

(a) (40pts) Read through the Value_Iteration.ipynb and implement value_iteration function. The stopping tolerance (defined as $max_s|V_{old}(s) - V_{new}(s)|$) is 10^{-3}. Use $\gamma = 1.0$. Return the optimal value function and optimal policy.

Solution: See Value_Iteration_Solution.ipynb

(b) (40pts) Read through the Policy_Iteration.ipynb and implement policy_improvement function. Policy evaluation is provided for convenience. The stopping tolerance is 10^{-4}. Use $\gamma = 1.0$. Return the optimal value function and optimal policy.

Solution: See Policy_Iteration_Solution.ipynb

[2] Adopted from https://github.com/dennybritz/reinforcement-learning