AI2

Assignment 4

Lazar Popov (s3340473)
Sander Kaatee (s3150798)
Group 19

October 24, 2020

1 Islands Problem

Iteration 0
Value-determination:
$U_{\pi_0}(s_1) \leftarrow 0 + 0.5 * ((0.5 * U_{\pi_0}(s_2)) + (0.5 * U_{\pi_0}(s_1)))$
$U_{\pi_0}(s_2) \leftarrow -1 + 0.5 * ((0.5 * U_{\pi_0}(s_3)) + (0.5 * U_{\pi_0}(s_2)))$
$U_{\pi_0}(s_3) \leftarrow 1$

$U_{\pi_0}(s_2) \leftarrow -1 + 0.5 * ((0.5 * 1) + (0.5 * -1)) = -1$

$U_{\pi_0}(s_1) \leftarrow 0 + 0.5 * ((0.5 * -1) + (0.5 * 0)) = -0.25$

We go through each of *state s* :
We find the action that will result in the best utility from state s_1:
State s_1:
$MAX[(T(s_1, a(s_1 \rightarrow s_2)) * U(s_2)), (T(s_1, a(s_1 \rightarrow s_3)) * U(s_3))]$
$MAX[((0.5 * -1) + (0.5 * -0.25)), ((0.5 * 1) + (0.5 * -0.25))] = 0.375$

if (Utility of best action at $s_1 > pi_0(s_1)$):
if $(0.375 > ((0.5 * -1) + (0.5 * -0.25))$:
if $(0.375 > -0.625)$:
if (TRUE):
    $\pi_1(s_1) \leftarrow a(s_1 \rightarrow s_3)$

State s_2:
We find the action that will result in the best utility from state s_2 $MAX[(T(s_2, a(s_2 \rightarrow s_1)) * U(s_1)), (T(s_2, a(s_2 \rightarrow s_3)) * U(s_3))]$
$MAX[((0.5 * -0.25) + (0.5 * -1)), ((0.5 * -1) + (0.5 * 1))] = 0$

if (Utility of best action at $s_2 > pi_0(s_2)$):
if $(0 > ((0.5 * -1) + (0.5 * 1))$:
if $(0 > 0)$:
if (FALSE):
    no new policy added

State s_3:
Agent will stay here, thus there are no actions to compare.

Iteration 1
Value-determination:
$U_{\pi_1}(s_1) \leftarrow 0 + 0.5 * ((0.5 * U_{\pi_1}(s_3)) + (0.5 * U_{\pi_1}(s_1)))$
$U_{\pi_1}(s_2) \leftarrow -1 + 0.5 * ((0.5 * U_{\pi_1}(s_3)) + (0.5 * U_{\pi_1}(s_2)))$
$U_{\pi_1}(s_3) \leftarrow 1$

$U_{\pi_1}(s_2) \leftarrow -1 + 0.5 * ((0.5 * 1) + (0.5 * -1)) = -1$

$U_{\pi_1}(s_1) \leftarrow 0 + 0.5 * ((0.5 * 1) + (0.5 * 0)) = 0.25$

We go through each of *state s* :
We find the action that will result in the best utility from state s_1*:*
State s_1:
$MAX[(T(s_1, a(s_1 \rightarrow s_2)) * U(s_2)), (T(s_1, a(s_1 \rightarrow s_3)) * U(s_3))]$
$MAX[((0.5 * -1) + (0.5 * 0.25)), ((0.5 * 1) + (0.5 * 0.25))] = 0.625$

if (Utility of best action at $s_1 > pi_1(s_1)$):
if $(0.625 > ((0.5 * 1) + (0.5 * 0.25))$:
if $(0.625 > 0.625)$:
if (FALSE):
    no new policy added

State s_2:
We find the action that will result in the best utility from state s_2 $MAX[(T(s_2, a(s_2 \rightarrow s_1)) * U(s_1)), (T(s_2, a(s_2 \rightarrow s_3)) * U(s_3))]$
$MAX[((0.5 * 0.25) + (0.5 * -1)), ((0.5 * -1) + (0.5 * 1))] = 0$

if (Utility of best action at $s_2 > pi_0(s_2)$):
if $(0 > ((0.5 * -1) + (0.5 * 1))$:
if $(0 > 0)$:
if (FALSE):
    no new policy added

State s_3:
Agent will stay here, thus there are no actions to compare.

Conclusion:
Therefore π_0 had the actions $(s_1 \rightarrow s_2)$ and $(s_2 \rightarrow s_3)$ in the states s_1 and s_2 respectively. After the policy iteration π_0 changed to π_1. π_1 now has the actions $(s_1 \rightarrow s_3)$ and $(s_2 \rightarrow s_3)$ in the states s_1 and s_2 respectively. Only the action from s_1 has changed.

5 Comparing our algorithms

- *Do both algorithms find the same solutions?*
  The two algorithms do not find the same solution while when the stop criterion is set to a bigger value (0.01). The reason for this is that the value iteration algorithm takes a riskier route when that is the case. When the stop criterion is a smaller value both algorithms behave similarly. The above statement is true for both the RN and 2D problem.

- *What is the number of iterations required for each algorithm to find a solution?*
  When γ is a bit lower the value iteration algorithm performs the best. For this question we are using $\gamma = 0.001$. For Value Iteration applied to the RN problem it takes 4 epochs to arrive to the best solution. For the Policy Iteration for the RN problem it takes on average

4 epochs as well to arrive to the optimal solution. Since we are using random initial policy the epoch it takes can be 3, 4, sometimes 5.

For the Value Iteration applied to the 2D problem it takes 16 epoch to arrive to the best solution. For the Policy Iteration applied to the 2D problem it takes on average 3 epoch to arrive the correct solution, again because of the random initial policy.

- *What is the time required for each algorithm to find a solution?*
  With a stop criterion (δ) of 0.01, the required time to find a solution for each algorithm is:

  ```
  valueIteration() = 0.0004277229309082031 seconds
  policyIteration() = 7.152557373046875e-07 seconds
  ```

  Do note that the time for `policyIteration` is printed as a scientific number, thus being about a 1000 times as fast in comparison to `valueIteration` for a stop criterion of 0.01

- *What is the influence of the discount factor on the way both algorithms perform?*
  Increasing the discount factor to one make the algorithms choose the safest path to the goals in both problems. This safest path might be longer but it decreases the chance of ending up to a state of -1. When we decrease the discount factor the algorithm chooses more dangerous route, meaning a route that is closer to states with -1. When the discount factor approaches zero, the algorithms might not ever reach the goal state. The reason for that is because when the utilities of the states are calculated the algorithm starts favouring their current state if neighboring states are contaminated by a nearby -1. This might make the algorithm decide that is better to move towards a wall and give itself a 90% chance to stay at the same state.

- *What is the influence of the stop criterion δ in value iteration?*
  The stop criterion is the threshold for the difference in utility for states per epoch. If the maximum difference in utility for all states between two epochs is lower than the stop criterion, then the algorithm is finished. This means that having a higher stop criterion results in the cutting of earlier, as the changes are bigger at the start and converge to zero with each repetition.
  Thus, a higher stop criterion results in a faster algorithm, while a lower stop criterion will find more attuned utilities. Whether this finer tuning also results in a better chain of decisions depends on several factors, like map-layout, reward-functions, discount factor, etc.

- *How do reinforcement learning algorithms such as Q-Learning find solutions for sequential decision problems without knowing the transition function explicitly*
  Reinforcement learning algorithms learn to find solutions to sequential decision problems by trying different <state,action> pairs and figuring out an expected reward.

  For example, Q-learning searches for optimal Q-values by initializing a table of Q values for each <state,action> pair at 0, e.g: $\hat{Q}(s, a) \leftarrow 0$. It then selects a random action, executes it, receives the reward and observes the new state and then updates the the table entry according to the formula:
  $\hat{Q}(\boldsymbol{s}, \boldsymbol{a}) \leftarrow \boldsymbol{r}(\boldsymbol{s}, \boldsymbol{a}) + \boldsymbol{y} \max_{a'} \hat{Q}\left(\boldsymbol{s}', \boldsymbol{a}'\right)$