University of Groningen

Artificial Intelligence 2

Lab 4: Value and Policy Iteration

Siger Steenstra
s2408791

October 25, 2016

1 Island Problem

The original policy is to go from state 1 to state 2 to state 3.
$\pi = ([S_1, S_2], [S_2, S_3])$
$R(S_1) = 0, R(S_2) = -1, R(S_3) = 1, \gamma = 0.5$
Utilities are then given by:
$U(S_1) = R(S_1) + \gamma * 0.5 * U(S_2)$
$U(S_2) = R(S_2) + \gamma * 0.5 * U(S_3)$
$U(S_3) = R(S_3) + \gamma * U(S_3)$

$U(S_1) = 0 + 0.25 * U(S_2)$
$U(S_2) = -1 + 0.25 * U(S_3)$
$U(S_3) = 1 + \gamma * U(S_3)$
$1 = \frac{1}{U(S_3)} + \gamma$
$1 - \gamma = \frac{1}{U(S_3)}$
$U(S_3) = \frac{1}{1-\gamma}$
$U(S_3) = \frac{1}{1-0.5} = 2$
Then:
$U(S_2) = -1 + 0.25 * 2 = -0.5$, and:
$U(S_1) = 0 + 0.25 * -0.5 = -0.125$
Then the policy is updated:
$U(S_1) = R(S_1) + 0.5 * max(0.5 * U(S_2), 0.5 * U(S_3))$
$U(S_1) = 0.5 * max(0.5 * -0.5, 0.5 * 2)$
$U(S_1) = 0.5 * max(-0.25, 1)$
$U(S_1) = 0.5 * 1 = 0.5$
Here we see that state 3 is the best choice, which does not agree with the current policy.
$U(S_2) = R(S_2) + 0.5 * max(0.5 * U(S_1), 0.5 * U(S_3))$
$U(S_2) = -1 + 0.5 * max(0.5 * -0.125, 0.5 * 2)$
$U(S_2) = -1 + 0.5 * max(-0.0625, 1)$
$U(S_2) = -1 + 0.5 * 1 = -0.5$
Also from state 2, state 3 is the best choice, which agrees with the current policy.
So now we have the policy:
$\pi = ([S_1, S_3], [S_2, S_3])$
With these newly calculated utilities, we can update the policy again and check whether it converges to a stable policy:
$U(S_1) = R(S_1) + 0.5 * max(0.5 * U(S_2), 0.5 * U(S_3))$
$U(S_1) = 0.5 * max(0.5 * -0.5, 0.5 * 2)$
$U(S_1) = 0.5 * max(-0.25, 1)$
$U(S_1) = 0.5 * 1 = 0.5$
So from state 1 it is still best to go to state 3.
$U(S_2) = R(S_2) + 0.5 * max(0.5 * U(S_1), 0.5 * U(S_3))$
$U(S_2) = -1 + 0.5 * max(0.5 * 0.5, 0.5 * 2)$
$U(S_2) = -1 + 0.5 * max(0.5, 1)$

$U(S_2) = -1 + 0.5 * 1 = -0.5$
So from state 2 it is also still best to go to state 3.
This leaves us with the policy:
$\pi = ([S_1, S_3], [S_2, S_3])$
Since this policy is exactly the same as the previous policy, it is safe to say it has converged to an optimal policy.

2 Value Iteration

The programming has once again been a collaboration between Davey Schilling (s2199041) and myself.

3 Testing

We have written scripts `exe.py` and `exe2.py` to make testing the code more time efficient. These scripts make the problem, perform the iterations and print the results. During programming these scripts were quite useful because they saved a lot of time.

```
$ python exe.py
Number of iterations for value iteration: 7.
:--------:--------:--------:--------:
|  0.300 |  0.472 |  0.682 |    1   |
:--------:--------:--------:--------:
|  0.180 |        |  0.344 |   -1   |
:--------:--------:--------:--------:
|  0.090 |  0.095 |  0.188 |  0.000 |
:--------:--------:--------:--------:

Number of iterations for policy iteration: 3.
:--------:--------:--------:--------:
|   >>   |   >>   |   >>   |    1   |
:--------:--------:--------:--------:
|   /\   |        |   /\   |   -1   |
:--------:--------:--------:--------:
|   /\   |   >>   |   /\   |   <<   |
:--------:--------:--------:--------:


$ python exe2.py
Number of iterations for value iteration: 11.
:--------:--------:--------:--------:--------:--------:--------:--------:--------:--------:
| -0.156 | -0.135 | -0.101 | -0.052 |  0.017 |  0.106 |  0.228 |  0.398 |  0.637 |    1   |
:--------:--------:--------:--------:--------:--------:--------:--------:--------:--------:
| -0.131 |        | -0.127 | -0.098 |        |        |        |        |  0.422 |  0.635 |
:--------:--------:--------:--------:--------:--------:--------:--------:--------:--------:
| -0.093 |        | -0.147 | -0.132 | -0.153 | -0.167 | -0.176 |        |  0.258 |  0.384 |
:--------:--------:--------:--------:--------:--------:--------:--------:--------:--------:
| -0.043 |        | -0.162 | -0.154 | -0.165 |        | -0.182 |        |  0.136 |  0.210 |
:--------:--------:--------:--------:--------:--------:--------:--------:--------:--------:
|  0.023 |        | -0.173 | -0.168 | -0.175 |        | -0.186 |        |  0.045 |  0.088 |
:--------:--------:--------:--------:--------:--------:--------:--------:--------:--------:
|  0.114 |        | -0.180 | -0.178 | -0.181 |        | -0.186 |        | -0.022 |  0.003 |
:--------:--------:--------:--------:--------:--------:--------:--------:--------:--------:
|  0.239 |        | -0.185 | -0.184 | -0.186 |        | -0.187 |        | -0.071 | -0.057 |
:--------:--------:--------:--------:--------:--------:--------:--------:--------:--------:
|  0.414 |        | -0.187 | -0.188 | -0.189 |        | -0.187 |        | -0.109 | -0.106 |
:--------:--------:--------:--------:--------:--------:--------:--------:--------:--------:
|  0.658 |        | -0.187 |        |        |        | -0.188 |        | -0.149 | -0.208 |
:--------:--------:--------:--------:--------:--------:--------:--------:--------:--------:
|    1   |        | -0.187 | -0.187 | -0.187 | -0.188 | -0.190 | -0.197 | -0.238 |   -1   |
:--------:--------:--------:--------:--------:--------:--------:--------:--------:--------:

Number of iterations for policy iteration: 5.
```

```
:--------:--------:--------:--------:--------:--------:--------:--------:--------:--------:
|   \/   |   >>   |   >>   |   >>   |   >>   |   >>   |   >>   |   >>   |   >>   |    1   |
:--------:--------:--------:--------:--------:--------:--------:--------:--------:--------:
|   \/   |        |   /\   |   /\   |        |        |        |        |   /\   |   /\   |
:--------:--------:--------:--------:--------:--------:--------:--------:--------:--------:
|   \/   |        |   /\   |   /\   |   <<   |   <<   |   <<   |        |   /\   |   /\   |
:--------:--------:--------:--------:--------:--------:--------:--------:--------:--------:
|   \/   |        |   /\   |   /\   |   /\   |        |   /\   |        |   /\   |   /\   |
:--------:--------:--------:--------:--------:--------:--------:--------:--------:--------:
|   \/   |        |   /\   |   /\   |   /\   |        |   /\   |        |   /\   |   /\   |
:--------:--------:--------:--------:--------:--------:--------:--------:--------:--------:
|   \/   |        |   /\   |   /\   |   /\   |        |   /\   |        |   /\   |   /\   |
:--------:--------:--------:--------:--------:--------:--------:--------:--------:--------:
|   \/   |        |   /\   |   /\   |   /\   |        |   /\   |        |   /\   |   /\   |
:--------:--------:--------:--------:--------:--------:--------:--------:--------:--------:
|   \/   |        |   /\   |   /\   |   /\   |        |   /\   |        |   /\   |   /\   |
:--------:--------:--------:--------:--------:--------:--------:--------:--------:--------:
|   \/   |        |   /\   |        |        |        |   /\   |        |   /\   |   /\   |
:--------:--------:--------:--------:--------:--------:--------:--------:--------:--------:
|    1   |        |   /\   |   <<   |   <<   |   <<   |   /\   |   <<   |   /\   |   -1   |
:--------:--------:--------:--------:--------:--------:--------:--------:--------:--------:
```

The solutions for both algorithms is the same for both problems as you can see above. The number of iterations is displayed in the table below:

# of iterations	valueIteration	policyIteration
RNP	7	3
2DP	11	5

Repeated executing of this code shows that in very few occurences the number of iterations is slightly different from the values represented above.

Decreasing the discount factor will lead to less reward for other states.
You can keep lowering the stop criterion to get more accurate rewards per state, but it will not change the solution. So the stop criterion should be low, but not too low, as it will not improve the final path, but it will take more iterations.

Reinforcement learning can make use of feedback about the choices it has made earlier to decide what is the best way to go. You do not need the transition probabilities because you know what the goal is and what the possibilities are to get there.