Exercise 2 - MDPs

Tim Schäfer, Timo Mahringer

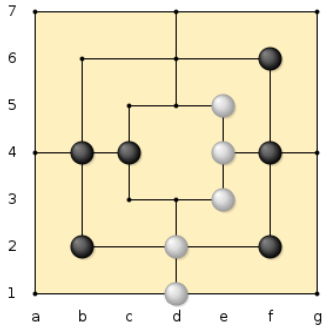
Exercise 1

a) Set of states is max. (8x8)^{13} in 3 dimensions. Actions are all possible moves. And the whole environment is discrete. Reward may be the number of figures (weighted eg. more for king) or the inverted number of figures of the opponent. We can also consider “saved/killed” figures in a move.

b) State space consists out of two continuous 6D-poses for the object and the endeffector. Action space is continuous in the task-space of the endeffector with its DOFs. Distance of the object to the goal pose.

c) State is the continuous 6D-pose of the drone. And actions is again the task-space of the drone. Reward may be the difference to the goal pose or/and the derivative of the pose should be small.

d) Nine men’s morris board game. With 18 stones (9 black/9 white) and 8x3 = 24 so the state space is max. 24x16=384. And Actions are all possible moves. Reward how many stones you got left.



Exercise 2:

a) Bandits don’t change any state and playing them does not change anything in the environment. So maximising each round on its own automatically maximises the future rewards. This is why future rewards do not have to be considered.

b) Using slides 31 and 33: Take the result of slide 33 and simply use this to replace the backmost sum of the Bellmann equation to get

$$v_\pi(s) = \sum_a \pi(a \,|\, s) q_\pi(s,a) = \sum_a \pi(a \mid s) \sum_{s',r} p(s',r \mid s,a)\Big[r+\gamma v_\pi(s')\Big] \text{ for all } s \in \mathcal{S}$$

$$q_\pi(s,a) = \sum_{s',r} p(s',r \mid s,a)\Big[r+\gamma \sum_{a'} \pi(a'|s') q_\pi(s',a')\Big]$$

$$v_\pi = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_\pi(s')\right]$$

c)

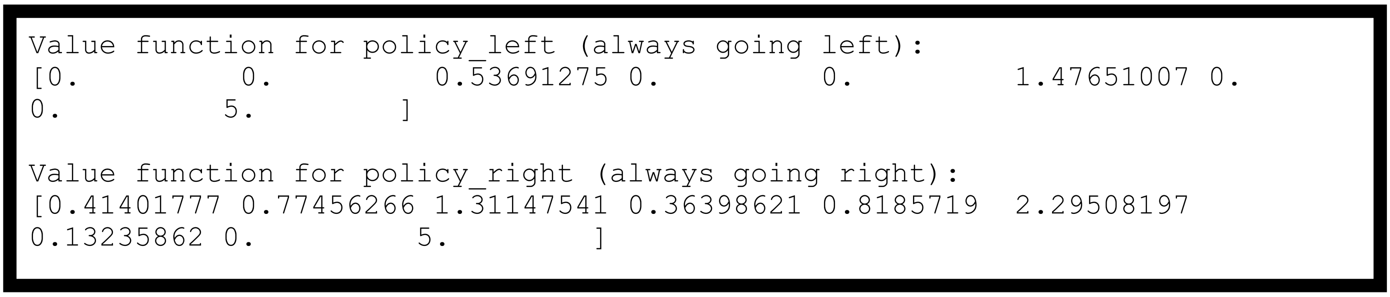
$$= \sum_a \pi(a|s) \left\{ \underbrace{\sum_r r \sum_{s'} p(s',r|s,a)}_{=r(s,a)\text{ see slide 22}} + \gamma \sum_{s'} v_\pi(s') \quad \underbrace{\sum_r p(s',r|s,a)}_{=p(s'|s,a)\text{ see slide 21}} \right\}$$

$$= \sum_a \pi(a|s) \left\{ r(s,a) + \gamma \sum_{s'} v_\pi(s') p(s'|s,a) \right\}$$

Exercise 3:

a) State space size is = 3x3 = 9 and Action space size is = 4
We can have 4^9 = 262,144 different possible policies.

b)



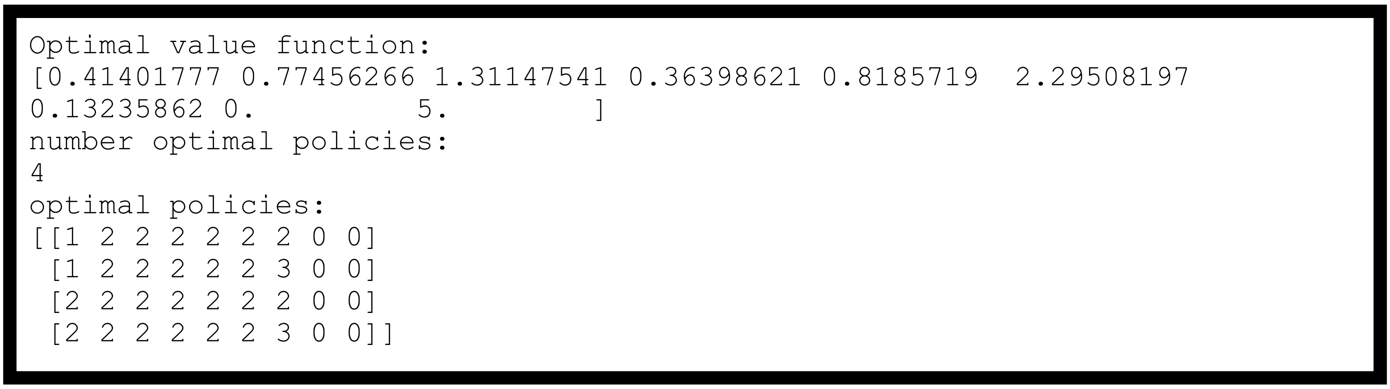
```
Value function for policy_left (always going left):
[0.         0.         0.53691275 0.         0.         1.47651007 0.
0.         5.        ]

Value function for policy_right (always going right):
[0.41401777 0.77456266 1.31147541 0.36398621 0.8185719  2.29508197
0.13235862 0.         5.        ]
```

$$v_\pi = r + \gamma P_\pi v_\pi$$

$$\left(\mathbf{1} - \gamma P_\pi\right) v_\pi = r$$

$$v_\pi = \left(\mathbf{1} - \gamma P_\pi\right)^{-1} r$$

c)



```
Optimal value function:
[0.41401777 0.77456266 1.31147541 0.36398621 0.8185719  2.29508197
0.13235862 0.         5.        ]
number optimal policies:
4
optimal policies:
[[1 2 2 2 2 2 2 0 0]
 [1 2 2 2 2 2 3 0 0]
 [2 2 2 2 2 2 2 0 0]
 [2 2 2 2 2 2 3 0 0]]
```

d) It should not work as well because of the growing number of possible policies.(Dynamic Programming) We assume that we can try as many policies as we want without changing the environment. This is not possible for most real environment.