Programming Session: Exercise

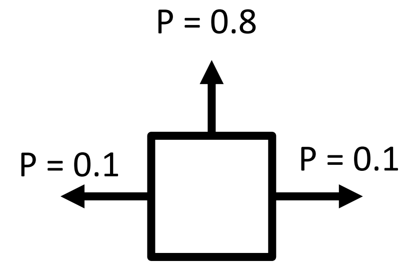
Exercise 1: Q-Learning

0	0	0	1
0		0	-100
0	0	0	0

Rewards r(s)

$\gamma = 0.9$

The agent moves in the selected direction with probability 0.8 and in the perpendicular directions with prob. 0.1. If the agent bumps the wall, it stays in the same cell.



Implement Q-Learning to find the optimal policy for the problem presented above.

Present the maximum Q-values at each state in a matrix form coinciding with the grid (3x4). Specify the action for each state from the final policy in a matrix form (3x4). Use the notation 1 for action up, 2 for down, 3 for left, and 4 for right.

Algorithm 2 Q-Learning

Initialize $Q(s,a)$
observe current state s
loop
  select action a in s according to exploration-exploitation strategy
  execute a, get $r(s,a)$, and observe new state s'
  estimate maximum $Qmax = \max_{a'} Q(s',a')$
  generate $q(s,a) = r(s,a) + \gamma Qmax$
  $Q(s,a) \Leftarrow Q(s,a) + \eta\,(q(s,a) - Q(s,a))$
  $s \Leftarrow s'$
end loop

Hint: Create a function `[snext,r] = simulator(s,a)` to execute selected actions.

Exercise 2: SARSA

Implement the SARSA algorithm to find the optimal policy for the problem presented in Exercise 1.

Present the Q-values for each action as well as the the maximum Q-value for each state in seprate matrices (total 4 matrices of 3x4).

Specify the action for each state from the final policy in a matrix form (3x4). Use the notation 1 for action up, 2 for down, 3 for left, and 4 for right.

Algorithm 1 SARSA

Initialize $Q(s, a)$
observe current state s
select action a in s according to current action policy
loop
  execute a, get $r(s, a)$, and observe new state s'
  choose a' in s' using, for instance, the ϵ-greedy strategy (policy improvement)
  generate $q(s, a) = r(s, a) + \gamma Q(s', a')$
  $Q(s, a) \Leftarrow Q(s, a) + \eta\,(q(s, a) - Q(s, a))$ (policy evaluation)
  $s \Leftarrow s'$
  $a \Leftarrow a'$
end loop

Programming Session

Implement the code (preferably) in Matlab. Save all the implemented files in a folder `L2_surnames`. Implement a script called `test.m` that executes the implemented functions and presents the requested matrices.

Send the folder compressed (.zip) by email to alejandro.agostini@tum.de with the subject RLRWS20_L2_surnames.