CS 229 - Machine Learning
Xiangliang Zhang
Computer Science (CS)/Statistics (STAT) Program
Computer, Electrical and Mathematical Sciences & Engineering (CEMSE) Division
King Abdullah University of Science and Technology (KAUST)

Homework X: Reinforcement Learning

Henrique Aparecido Laureano

Spring Semester 2018

Contents

Question 1: [70 points] Optimal values and policy

Given the grid world in Figure 1, there are 4 deterministic actions: *up*, *down*, *left* and *right*. The goal is to reach the G, starting at S.

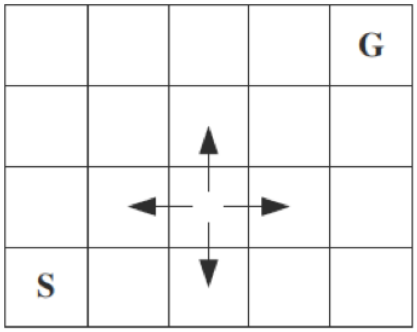

Figure 1: Grid world.

The reward on reaching on the goal (G) is 100. The reward on actions that would take the agent off the grid is -1 (agent stays still in this case). The reward on other actions is 0. The discount factor $\gamma = 0.9$.

Use Q learning to learn the optimal values of $Q^*(s, a)$. Please submit your own code on calculating the $Q^*(s, a)$.

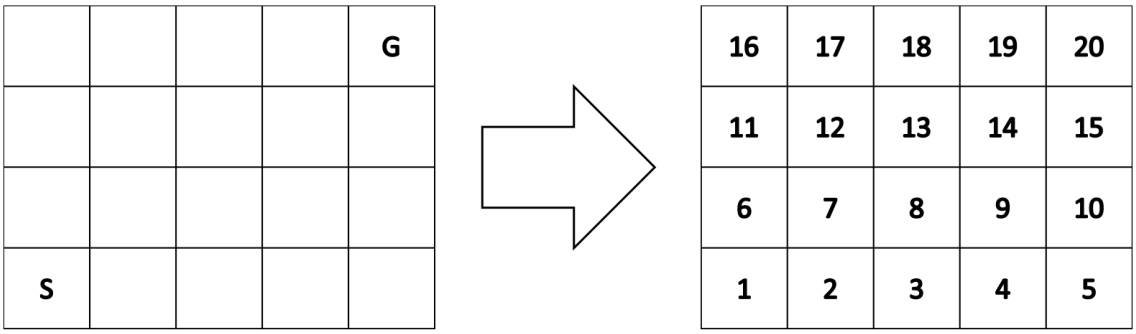

Figure 2: Attributing numbers (names) to the states.

By the rewards previously defined we can build the R matrix, given by the following code and presented in Table 1.

```r
# <r code> ============================================================================ #
R <- matrix(
  c(-1,  0, -1, -1, -1,  0, rep(-1, 14), # ---------------------------------- 1
     0, -1,  0, -1, -1, -1,  0, rep(-1, 13), # ------------------------------- 2
    -1,  0, -1,  0, -1, -1, -1,  0, rep(-1, 12), # ---------------------------- 3
    -1, -1,  0, -1,  0, -1, -1, -1,  0, rep(-1, 11), # ----------------------- 4
    -1, -1, -1,  0, -1, -1, -1, -1, -1,  0, rep(-1, 10), # ------------------- 5
     0, -1, -1, -1, -1, -1,  0, -1, -1, -1,  0, rep(-1, 9), # ---------------- 6
    -1,  0, -1, -1, -1,  0, -1,  0, -1, -1, -1,  0, rep(-1, 8), # ------------ 7
    -1, -1,  0, -1, -1, -1,  0, -1,  0, -1, -1, -1,  0, rep(-1, 7), # -------- 8
    -1, -1, -1,  0, -1, -1, -1,  0, -1,  0, -1, -1, -1,  0, rep(-1, 6), # --- 9
    rep(-1, 4),  0, -1, -1, -1,  0, -1, -1, -1, -1, -1,  0, rep(-1, 5), # -- 10
    rep(-1, 5),  0, -1, -1, -1, -1, -1,  0, -1, -1, -1,  0, rep(-1, 4), # -- 11
    rep(-1, 6),  0, -1, -1, -1,  0, -1,  0, -1, -1, -1,  0, -1, -1, -1, # -- 12
    rep(-1, 7),  0, -1, -1, -1,  0, -1,  0, -1, -1, -1,  0, -1, -1, # ------ 13
    rep(-1, 8),  0, -1, -1, -1,  0, -1,  0, -1, -1, -1,  0, -1, # ---------- 14
    rep(-1, 9),  0, -1, -1, -1,  0, -1, -1, -1, -1, -1, 100, # ------------- 15
    rep(-1, 10), 0, -1, -1, -1, -1, -1,  0, -1, -1, -1, # ------------------ 16
    rep(-1, 11), 0, -1, -1, -1,  0, -1,  0, -1, -1, # ---------------------- 17
    rep(-1, 12), 0, -1, -1, -1,  0, -1,  0, -1, # -------------------------- 18
    rep(-1, 13), 0, -1, -1, -1,  0, -1, 100, # ------------------------------ 19
    rep(-1, 14), 0, -1, -1, -1,  0, 100), nrow = 20, byrow = TRUE) # ---------- 20
# </r code> ============================================================================ #
```

Table 1: Reward table. In the rows we have the states, in the columns we have the actions.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
1	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
2	0	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
3	-1	0	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
4	-1	-1	0	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
5	-1	-1	-1	0	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
6	0	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1
7	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1
8	-1	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1
9	-1	-1	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1
10	-1	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1
11	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	-1
12	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1	0	-1	-1	-1
13	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1	0	-1	-1
14	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1	0	-1
15	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	100
16	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	0	-1	-1	-1
17	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	0	-1	-1
18	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	0	-1
19	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	100
20	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	100

Implementing Q learning

```
# <r code> ============================================================================ #
q.learn <- function(R, iter, gamma, goal) {
  # this algorithm is build for one or two goal states ------------------------- #
  # if we have only one, replicate the goal state ------------------------------ #
  # below, in the while loop break out criterion you can see the why ----------- #
  if (length(goal) == 1) goal[2] = 1
  # initialize Q to be zero matrix, same size as R ----------------------------- #
  Q = matrix(rep(0, length(R)), nrow = nrow(R))
  # initial state is 1, S, as previously defined ------------------------------- #
  # cs: current state ---------------------------------------------------------- #
  cs = 1
  # loop over episodes --------------------------------------------------------- #
  for (i in 1:iter) {
    # iterate until we get to the goal state ----------------------------------- #
    while (1) {
      # choose next state from possible actions at current state --------------- #
      next.states = which(R[cs, ] > -1)
      # if only one possible action, then choose it ---------------------------- #
      if (length(next.states) == 1)
        ns = next.states
      # otherwise, choose one at random ---------------------------------------- #
      else
        ns = sample(next.states, 1)
      # updating --------------------------------------------------------------- #
      Q[cs, ns] = R[cs, ns] + gamma * max(Q[ns, which(R[ns ,] > -1)])
      # break out of while loop if goal state is reached ----------------------- #
      if (ns == goal[1] | ns == goal[2]) break
      # otherwise, set next.state as current state and repeat ------------------ #
      cs = ns
    }
    # for each episode, choose an initial state at random ---------------------- #
    cs = sample(1:nrow(R), 1)
  }
  return(100 * Q / max(Q)) # return resulting Q normalized by max value -------- #
}
# </r code> ============================================================================ #
```

(50pts) What is the $Q^*(s, a)$ for each pair of s and a?

Running two thousand, 2e3, iterations:

```
# <r code> ============================================================================ #
Q <- q.learn(R, iter = 2e3, gamma = .9, goal = 20)
# </r code> ============================================================================ #
```

The $Q^*(s, a)$ for each pair of s and a is given by the code above and presented in the following Table 2.

Table 2: Q* matrix/table. In the rows we have the states, in the columns we have the actions.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
1	0	52	0	0	0	52	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	47	0	59	0	0	0	59	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	53	0	66	0	0	0	66	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	59	0	73	0	0	0	73	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	66	0	0	0	0	0	81	0	0	0	0	0	0	0	0	0	0
6	47	0	0	0	0	0	59	0	0	0	59	0	0	0	0	0	0	0	0	0
7	0	53	0	0	0	53	0	66	0	0	0	66	0	0	0	0	0	0	0	0
8	0	0	59	0	0	0	59	0	73	0	0	0	73	0	0	0	0	0	0	0
9	0	0	0	66	0	0	0	66	0	81	0	0	0	81	0	0	0	0	0	0
10	0	0	0	0	73	0	0	0	73	0	0	0	0	0	90	0	0	0	0	0
11	0	0	0	0	0	53	0	0	0	0	0	66	0	0	0	66	0	0	0	0
12	0	0	0	0	0	0	59	0	0	0	59	0	73	0	0	0	73	0	0	0
13	0	0	0	0	0	0	0	66	0	0	0	66	0	81	0	0	0	81	0	0
14	0	0	0	0	0	0	0	0	73	0	0	0	73	0	90	0	0	0	90	0
15	0	0	0	0	0	0	0	0	0	81	0	0	0	81	0	0	0	0	0	100
16	0	0	0	0	0	0	0	0	0	0	59	0	0	0	0	0	73	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	66	0	0	0	66	0	81	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	73	0	0	0	73	0	90	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	81	0	0	0	81	0	100
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	90	0	0	0	90	100

□

(10pts) What is the $V^*(s)$ for each s?

The optimal policies $V^*(s)$ is defined by

$$V^*(s) = \max_{a \in A(s)} Q^*(s).$$

Therefore, looking to $Q^*(s)$ presented in Table 2 we easily see the $V^*(s)$ for each state s. We present this in the following Table 3.

Table 3: $V^*(s)$ for each s.

s:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
a:	2 or 6	3 or 7	4 or 8	5 or 9	10	7 or 11	8 or 12	9 or 13	10 or 14	15
$V^*(s)$:	52	59	66	73	81	59	66	73	81	90
s:	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
a:	12 or 16	13 or 17	14 or 18	15 or 19	20	17	18	19	20	20
$V^*(s)$:	66	73	81	90	100	73	81	90	100	100

We can see that for several states the $\max_{a \in A(s)} Q^*(s)$ is reached with two states, in this situations one of both can be chosen randomly.

□

(10pts) What are the actions of optimal policy?

Starting in **S**, represented as state 1, the maximum Q values suggest two alternatives: go to state 2 or 6. Suppose we arbitrarily choose to go to 2.

From state 2 the maximum Q values suggest two alternatives: go to state 3 or 7. Suppose we arbitrarily choose to go to 7.

From state 7 the maximum Q values suggest two alternatives: go to state 8 or 12. Suppose we arbitrarily choose to go to 8.

From state 8 the maximum Q values suggest two alternatives: go to state 9 or 13. Suppose we arbitrarily choose to go to 13.

From state 13 the maximum Q values suggest two alternatives: go to state 14 or 18. Suppose we arbitrarily choose to go to 14.

From state 14 the maximum Q values suggest two alternatives: go to state 15 or 19. Suppose we arbitrarily choose to go to 19.

From state 19 the maximum Q values suggests the action to go to state 20.

Thus, here the sequence is: 1 (**S**) - 2 - 7 - 8 - 13 - 14 - 19 - 20 (**G**). A representation is given below in Figure 3.

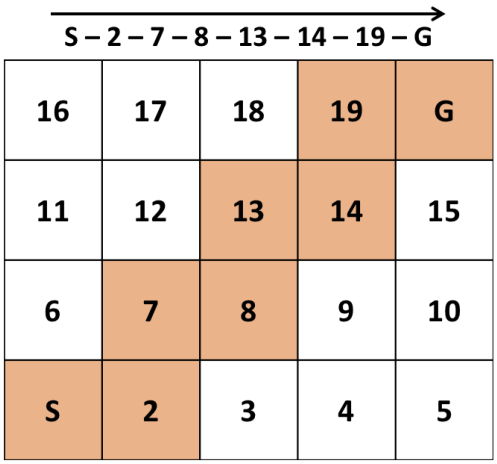

Figure 3: Representation of the actions for the optimal policy.

□

Question 2: [30pts] A change on the gird world

(15pts) Add another goal state to the lower-right corner.

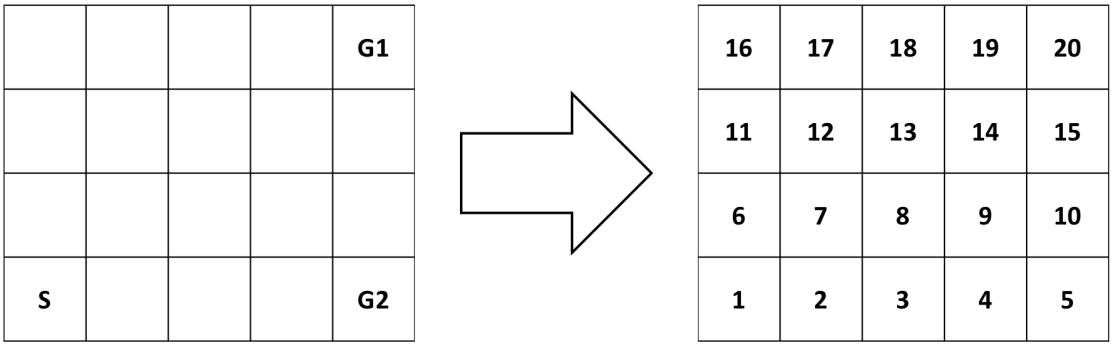

Figure 4: Attributing numbers (names) to the states.

How does the optimal policy change?

Now we add a reward of 100 on reaching on the (new) goal state 5. The new R matrix is given by the following code and presented in Table 4.

```
# <r code> ============================================================================ #
R <- matrix(
  c(-1,  0, -1,  -1,  -1,  0, rep(-1, 14), # --------------------------------- 1
     0, -1,  0,  -1,  -1, -1,  0, rep(-1, 13), # ------------------------------ 2
    -1,  0, -1,   0,  -1, -1, -1,  0, rep(-1, 12), # --------------------------- 3
    -1, -1,  0,  -1, 100, -1, -1, -1,   0, rep(-1, 11), # ---------------------- 4
    -1, -1, -1,   0, 100, -1, -1, -1,  -1,   0, rep(-1, 10), # ----------------- 5
     0, -1, -1,  -1,  -1, -1,  0, -1,  -1,  -1,  0, rep(-1, 9), # -------------- 6
    -1,  0, -1,  -1,  -1,  0, -1,  0,  -1,  -1, -1,  0, rep(-1, 8), # ---------- 7
    -1, -1,  0,  -1,  -1, -1,  0, -1,   0,  -1, -1, -1,  0, rep(-1, 7), # ------ 8
    -1, -1, -1,   0,  -1, -1, -1,  0,  -1,   0, -1, -1, -1,   0, rep(-1, 6), # - 9
    rep(-1, 4), 100,  -1, -1, -1,  0,  -1,  -1, -1, -1, -1,   0, rep(-1, 5), #  10
    rep(-1, 5),   0,  -1, -1, -1, -1,  -1,   0, -1, -1, -1,   0, rep(-1, 4), #  11
    rep(-1, 6),   0,  -1, -1, -1,  0,  -1,   0, -1, -1, -1,   0, -1, -1, -1, #  12
    rep(-1, 7),   0,  -1, -1, -1,  0,  -1,   0, -1, -1, -1,   0, -1, -1, # ---- 13
    rep(-1, 8),   0,  -1, -1, -1,  0,  -1,   0, -1, -1, -1,   0, -1, # -------- 14
    rep(-1, 9),   0,  -1, -1, -1,  0,  -1,  -1, -1, -1, -1, 100, # ------------ 15
    rep(-1, 10),  0,  -1, -1, -1, -1,  -1,   0, -1, -1, -1, # ----------------- 16
    rep(-1, 11),  0,  -1, -1, -1,  0,  -1,   0, -1, -1, # --------------------- 17
    rep(-1, 12),  0,  -1, -1, -1,  0,  -1,   0, -1, # ------------------------- 18
    rep(-1, 13),  0,  -1, -1, -1,  0,  -1, 100, # --------------------------- 19
    rep(-1, 14),  0,  -1, -1, -1,  0, 100), nrow = 20, byrow = TRUE) # -------- 20
# </r code> ============================================================================ #
```

Table 4: Reward table. In the rows we have the states, in the columns we have the actions.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
1	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
2	0	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
3	-1	0	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
4	-1	-1	0	-1	100	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
5	-1	-1	-1	0	100	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
6	0	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1
7	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1
8	-1	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1
9	-1	-1	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1
10	-1	-1	-1	-1	100	-1	-1	-1	0	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1
11	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	-1
12	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1	0	-1	-1	-1
13	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1	0	-1	-1
14	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1	0	-1
15	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	100
16	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	0	-1	-1	-1
17	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	0	-1	-1
18	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	0	-1
19	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	100
20	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	100

Running the Q learning algorithm two thousand, 2e3, iterations:

```
# <r code> ============================================================================ #
Q <- q.learn(R, iter = 2e3, gamma = .9, goal = c(5, 20))
# </r code> ============================================================================ #
```

The $Q^*(s, a)$ for each pair of s and a is given by the code above and presented in the following Table 5. Through Table 5 we are able to see which is the new optimal policy.

Starting in **S**, represented as state 1, the maximum Q values suggests the action to go to state 2.

From state 2 the maximum Q values suggests the action to go to state 3.

From state 3 the maximum Q values suggests the action to fo to state 4.

From state 4 the maximum Q values suggests the action to go to state 5, a goal state.

Thus, here the sequence is: 1 (**S**) - 2 - 3 - 4 - 5 (**G2**). A representation is given in the left hand side of the Figure 5.

Table 5: Q^* matrix/table. In the rows we have the states, in the columns we have the actions.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
1	0	73	0	0	0	59	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	66	0	81	0	0	0	65	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	73	0	90	0	0	0	73	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	81	0	100	0	0	0	81	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	89	99	0	0	0	0	90	0	0	0	0	0	0	0	0	0	0
6	65	0	0	0	0	0	66	0	0	0	59	0	0	0	0	0	0	0	0	0
7	0	73	0	0	0	59	0	72	0	0	0	65	0	0	0	0	0	0	0	0
8	0	0	81	0	0	0	65	0	81	0	0	0	73	0	0	0	0	0	0	0
9	0	0	0	90	0	0	0	73	0	90	0	0	0	81	0	0	0	0	0	0
10	0	0	0	0	100	0	0	0	81	0	0	0	0	0	90	0	0	0	0	0
11	0	0	0	0	0	59	0	0	0	0	0	65	0	0	0	65	0	0	0	0
12	0	0	0	0	0	0	65	0	0	0	59	0	73	0	0	0	73	0	0	0
13	0	0	0	0	0	0	0	73	0	0	0	65	0	81	0	0	0	81	0	0
14	0	0	0	0	0	0	0	0	81	0	0	0	73	0	90	0	0	0	90	0
15	0	0	0	0	0	0	0	0	0	90	0	0	0	81	0	0	0	0	0	100
16	0	0	0	0	0	0	0	0	0	0	59	0	0	0	0	0	73	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	65	0	0	0	65	0	81	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	73	0	0	0	73	0	90	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	81	0	0	0	81	0	100
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	90	0	0	0	90	99

Following the maximum Q values, to reach the goal state 20 (**G1**) we need/have to start from the state 11 or above (12, 13, ...). Starting from a state smaller than this we reach the state 5.

To ilustrate we have Figure 5. In the left we start in **S** (but we reach **G2** starting by any other state ≤ 10). In the right we start in 11 (but we reach **G1** starting by any state > 10).

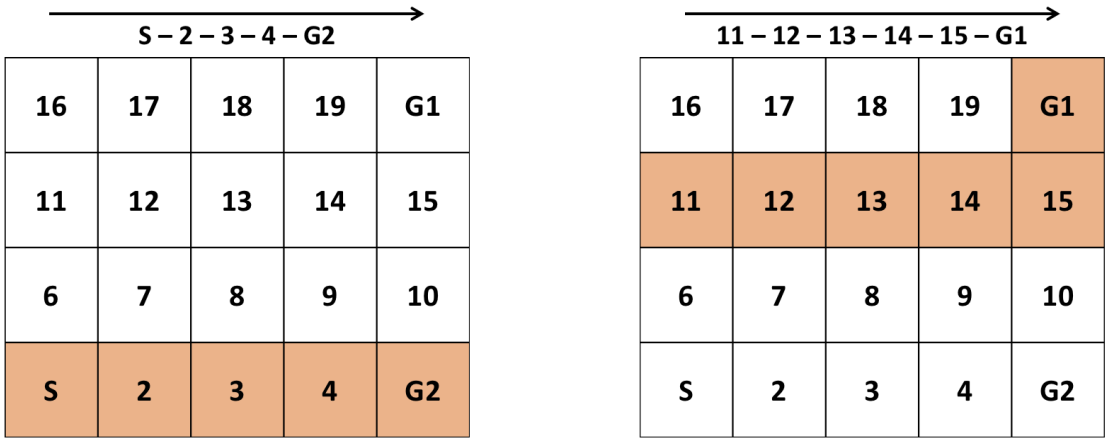

Figure 5: Representation of the actions for the optimal policy with two examples. In the left we start from a state ≤ 10, in the right we start from a state > 10.

□

(15pts) If a state of reward -100 (a very bad state) is defined in the lower-right corner. How does the optimal policy change?

First we modify the last R matrix putting a minus in the reward of the lower-right corner state. The new R matrix is given by the following code and presented in Table 6.

```
# <r code> ============================================================================ #
R <- matrix(
  c(-1,  0, -1,   -1,   -1,  0, rep(-1, 14), # ------------------------------- 1
     0, -1,  0,   -1,   -1, -1,  0, rep(-1, 13), # ---------------------------- 2
    -1,  0, -1,    0,   -1, -1, -1,  0, rep(-1, 12), # ------------------------ 3
    -1, -1,  0,   -1, -100, -1, -1, -1,   0, rep(-1, 11), # ------------------- 4
    -1, -1, -1,    0, -100, -1, -1, -1,  -1,   0, rep(-1, 10), # -------------- 5
     0, -1, -1,   -1,   -1, -1,  0, -1,  -1,  -1,  0, rep(-1, 9), # ------------ 6
    -1,  0, -1,   -1,   -1,  0, -1,  0,  -1,  -1, -1,  0, rep(-1, 8), # -------- 7
    -1, -1,  0,   -1,   -1, -1,  0, -1,   0,  -1, -1, -1,  0, rep(-1, 7), # ---- 8
    -1, -1, -1,    0,   -1, -1, -1,  0,  -1,   0, -1, -1, -1,   0, rep(-1, 6), # 9
    rep(-1, 4), -100,   -1, -1, -1,  0,  -1,  -1, -1, -1, -1,   0, rep(-1, 5), #10
    rep(-1, 5),    0,   -1, -1, -1, -1,  -1,   0, -1, -1, -1,   0, rep(-1, 4), #11
    rep(-1, 6),    0,   -1, -1, -1,  0,  -1,   0, -1, -1, -1,   0, -1, -1, -1, #12
    rep(-1, 7),    0,   -1, -1, -1,  0,  -1,   0, -1, -1, -1,   0, -1, -1, # -- 13
    rep(-1, 8),    0,   -1, -1, -1,  0,  -1,   0, -1, -1, -1,   0, -1, # ------ 14
    rep(-1, 9),    0,   -1, -1, -1,  0,  -1,  -1, -1, -1, -1, 100, # ---------- 15
    rep(-1, 10),   0,   -1, -1, -1, -1,  -1,   0, -1, -1, -1, # --------------- 16
    rep(-1, 11),   0,   -1, -1, -1,  0,  -1,   0, -1, -1, # ------------------ 17
    rep(-1, 12),   0,   -1, -1, -1,  0,  -1,   0, -1, # ---------------------- 18
    rep(-1, 13),   0,   -1, -1, -1,  0,  -1, 100, # -------------------------- 19
    rep(-1, 14),   0,   -1, -1, -1,  0, 100), nrow = 20, byrow = TRUE) # ------ 20
# </r code> ============================================================================ #
```

Running the Q learning algorithm two thousand, 2e3, iterations:

```
# <r code> ============================================================================ #
Q <- q.learn(R, iter = 2e3, gamma = .9, goal = c(5, 20))
# </r code> ============================================================================ #
```

The $Q^*(s, a)$ for each pair of s and a is given by the code above and presented in the following Table 7. Through Table 7 we are able to see which is the new optimal policy.

With a reward of -100 we'll never have the suggestion of go to the state 5 (**G2**), given by the maximum Q values. i.e., defining a very bad state, a long term is like if this state doesn't exist. Therefore, the optimal policy here is exactly the same that the one obtained in Table 2, Table 3 and Figure 3 (i.e., considering only one goal state).

The obtained $Q^*(s, a)$ for this state with high and negative reward are all zeros, as we can see in Table 7. A representation of the optimal policy is, therefore, the same that the one presented in Figure 3.

Table 6: Reward table. In the rows we have the states, in the columns we have the actions.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
1	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
2	0	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
3	-1	0	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
4	-1	-1	0	-1	-100	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
5	-1	-1	-1	0	-100	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
6	0	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1
7	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1
8	-1	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1
9	-1	-1	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	-1
10	-1	-1	-1	-1	-100	-1	-1	-1	0	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1
11	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	-1
12	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1	0	-1	-1	-1
13	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1	0	-1	-1
14	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1	0	-1
15	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	-1	-1	-1	-1	100
16	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	0	-1	-1	-1
17	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	0	-1	-1
18	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	0	-1
19	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	-1	100
20	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	0	100

Table 7: Q^* matrix/table. In the rows we have the states, in the columns we have the actions.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
1	0	53	0	0	0	53	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	48	0	59	0	0	0	59	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	53	0	66	0	0	0	66	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	59	0	0	0	0	0	73	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	65	0	0	0	0	0	81	0	0	0	0	0	0	0	0	0	0
6	48	0	0	0	0	0	59	0	0	0	59	0	0	0	0	0	0	0	0	0
7	0	53	0	0	0	53	0	66	0	0	0	66	0	0	0	0	0	0	0	0
8	0	0	59	0	0	0	59	0	73	0	0	0	73	0	0	0	0	0	0	0
9	0	0	0	66	0	0	0	66	0	81	0	0	0	81	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	73	0	0	0	0	0	90	0	0	0	0	0
11	0	0	0	0	0	53	0	0	0	0	0	65	0	0	0	65	0	0	0	0
12	0	0	0	0	0	0	59	0	0	0	59	0	73	0	0	0	73	0	0	0
13	0	0	0	0	0	0	0	66	0	0	0	66	0	81	0	0	0	81	0	0
14	0	0	0	0	0	0	0	0	73	0	0	0	73	0	90	0	0	0	90	0
15	0	0	0	0	0	0	0	0	0	81	0	0	0	81	0	0	0	0	0	100
16	0	0	0	0	0	0	0	0	0	0	59	0	0	0	0	0	73	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	66	0	0	0	65	0	81	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	73	0	0	0	73	0	90	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	81	0	0	0	81	0	100
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	90	0	0	0	90	100

■