Exercise 2

Eugeniu Vezeteu - 886240
ELEC-E8125 - Reinforcement Learning

September 21, 2020

1 Question 1 - What is the agent and the environment in this setup?

The agent is the sailor (boat), whose purpose is to reach the harbour. The environment is the sea, rocks, narrow passage between the rocks, wind and harbour.

2 Task 1 - Value iteration

The values are updated from the harbour to all the cells in the grid, closer to the harbour the higher the cell value is, see 1

r=0.00 V=0.53	r=0.00 V=0.59	r=0.00 V=0.66	r=0.00 V=0.73	r=0.00 V=0.66	r=-2.00 V=0.00	r=-2.00 V=0.00	r=-2.00 V=0.00	r=-2.00 V=0.00	r=-2.00 V=0.00	r=-2.00 V=0.00	r=0.00 V=7.72	r=0.00 V=8.71	r=0.00 V=9.84	r=10.00 V=0.00
r=0.00 V=0.59	r=0.00 V=0.66	r=0.00 V=0.74	r=0.00 V=0.83	r=0.00 V=0.80	r=-2.00 V=0.00	r=-2.00 V=0.00	r=-2.00 V=0.00	r=-2.00 V=0.00	r=-2.00 V=0.00	r=-2.00 V=0.00	r=0.00 V=6.95	r=0.00 V=7.80	r=0.00 V=8.76	r=0.00 V=9.84
r=0.00 V=0.65	r=0.00 V=0.73	r=0.00 V=0.83	r=0.00 V=0.94	r=0.00 V=1.06	r=0.00 V=1.21	r=0.00 V=1.74	r=0.00 V=2.39	r=0.00 V=3.21	r=0.00 V=4.20	r=0.00 V=5.59	r=0.00 V=6.19	r=0.00 V=6.95	r=0.00 V=7.80	r=0.00 V=8.71
r=0.00 V=0.64	r=0.00 V=0.72	r=0.00 V=0.81	r=0.00 V=0.90	r=0.00 V=1.01	r=0.00 V=1.14	r=0.00 V=1.64	r=0.00 V=2.24	r=0.00 V=2.97	r=0.00 V=3.84	r=0.00 V=4.98	r=0.00 V=5.52	r=0.00 V=6.19	r=0.00 V=6.95	r=0.00 V=7.72
r=0.00 V=0.65	r=0.00 V=0.73	r=0.00 V=0.81	r=0.00 V=0.90	r=0.00 V=0.82	r=-2.00 V=0.00	r=-2.00 V=0.00	r=-2.00 V=0.00	r=-2.00 V=0.00	r=-2.00 V=0.00	r=-2.00 V=0.00	r=0.00 V=4.91	r=0.00 V=5.51	r=0.00 V=6.18	r=0.00 V=6.84
r=0.00 V=0.73	r=0.00 V=0.82	r=0.00 V=0.92	r=0.00 V=1.02	r=0.00 V=1.00	r=-2.00 V=0.00	r=-2.00 V=0.00	r=-2.00 V=0.00	r=-2.00 V=0.00	r=-2.00 V=0.00	r=-2.00 V=0.00	r=0.00 V=4.37	r=0.00 V=4.91	r=0.00 V=5.50	r=0.00 V=6.06
r=0.00 V=0.81	r=0.00 V=0.91	r=0.00 V=1.03	r=0.00 V=1.16	r=0.00 V=1.30	r=0.00 V=1.47	r=0.00 V=1.66	r=0.00 V=1.88	r=0.00 V=2.14	r=0.00 V=2.61	r=0.00 V=3.19	r=0.00 V=3.89	r=0.00 V=4.37	r=0.00 V=4.89	r=0.00 V=5.37
r=0.00 V=0.89	r=0.00 V=1.00	r=0.00 V=1.13	r=0.00 V=1.28	r=0.00 V=1.45	r=0.00 V=1.65	r=0.00 V=1.87	r=0.00 V=2.11	r=0.00 V=2.40	r=0.00 V=2.72	r=0.00 V=3.08	r=0.00 V=3.47	r=0.00 V=3.89	r=0.00 V=4.35	r=0.00 V=4.76
r=0.00 V=0.86	r=0.00 V=0.96	r=0.00 V=1.08	r=0.00 V=1.22	r=0.00 V=1.37	r=0.00 V=1.54	r=0.00 V=1.73	r=0.00 V=1.94	r=0.00 V=2.18	r=0.00 V=2.45	r=0.00 V=2.75	r=0.00 V=3.09	r=0.00 V=3.47	r=0.00 V=3.87	r=0.00 V=4.22
r=0.00 V=0.81	r=0.00 V=0.90	r=0.00 V=1.01	r=0.00 V=1.13	r=0.00 V=1.27	r=0.00 V=1.42	r=0.00 V=1.58	r=0.00 V=1.77	r=0.00 V=1.98	r=0.00 V=2.21	r=0.00 V=2.47	r=0.00 V=2.76	r=0.00 V=3.09	r=0.00 V=3.44	r=0.00 V=3.74

Figure 1: Task 1 - Value iteration for the sailor example after 100 iterations

$V_{k+1}(s) = max_a E[R_{t+1} + \gamma * V_k(s_{t+1}|s_t = s, a_t = a] = max_a \sum_{s',r} p(s', r|s, a)[r + \gamma * V_k(s')]$ from [1].

```
1
2  def Task1(value_est,policy):
3      for iter in range(100):
4          env.clear_text()
5          #get all states
6          for i in range(env.w):
7              for j in range(env.h):
8                  history = []
9                  #compute the sum for all s'
10                 for tran in env.transitions[i,j]:
11                     sum = 0.
12                     for s_,r,done,p in tran:
13                         sum += p * (r + (gamma * value_est[s_
                               [0],s_[1]] if not done else 0))
14
15                     history.append(sum)
16
17                 #update V(s), and pi(s)
18                 value_est[i,j] = np.max(history)
19                 policy[i,j] = np.argmax(history)
20
21     return value_est,policy
```

3 Question 2 - What is the state value of the harbour and rock states? Why?

The state value of the harbour and the rock states is 0. The value of the state is expected sum of future rewards, starting from that state and following policy π, that's why for harbour state value is zero (there are no more rewards to get) and for rock states (we never start from this state).

4 Task 2 - optimal policy

The sailor reaches the harbour most of the times, see Figure 2 On this task policy computed in the previous task was used.

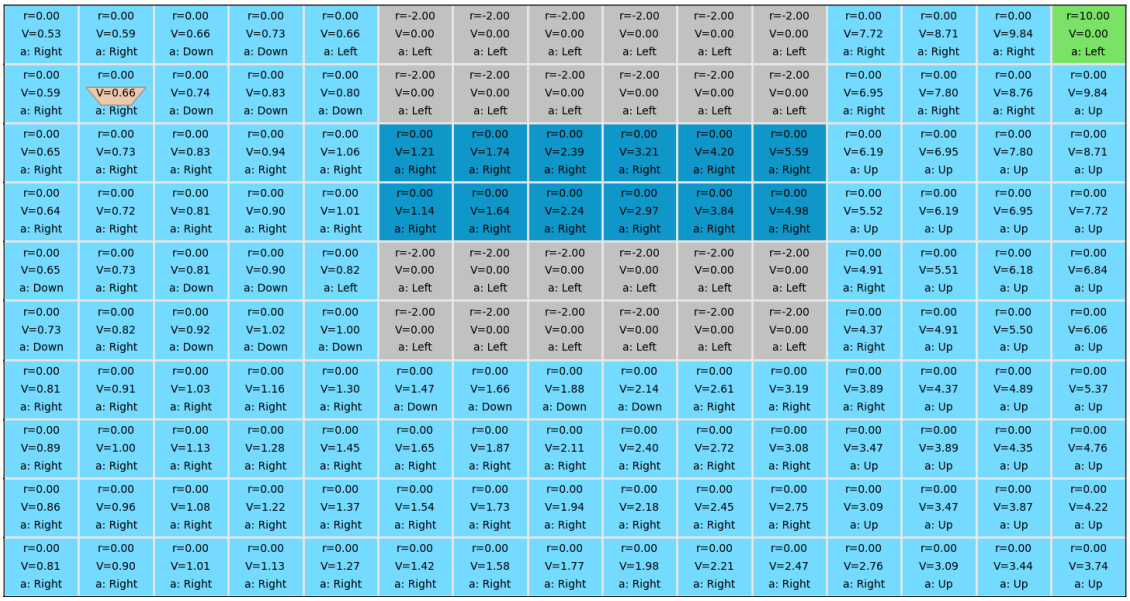

Figure 2: Task 2 - Value iteration, Action and reward

```
    for episod in range(10):
        done = False
        while not done:
            # TODO: Use the policy to take the optimal action (
                Task 2)
            action = policy[state]

            # Step the environment
            state, reward, done, _ = env.step(action)

            # Render and sleep
            env.render()
            sleep(0.1)
```

5 Question 3 - Which path did the sailor choose?

With penalty of -2, the sailor take the dangerous path between the rocks (with is the shortest), but when changing the penalty to -10, the agent took the safe path below the rocks. We can notice from Figure 3 that the state values between the rocks are much smaller then in Figure 2.

r=0.00 V=0.41 a: Down	r=0.00 V=0.45 a: Down	r=0.00 V=0.50 a: Down	r=0.00 V=0.54 a: Down	r=0.00 V=0.48 a: Left	r=-10.00 V=0.00 a: Left	r=-10.00 V=0.00 a: Left	r=-10.00 V=0.00 a: Left	r=-10.00 V=0.00 a: Left	r=-10.00 V=0.00 a: Left	r=-10.00 V=0.00 a: Left	r=0.00 V=7.72 a: Right	r=0.00 V=8.71 a: Right	r=0.00 V=9.84 a: Right	r=10.00 V=0.00 a: Left
r=0.00 V=0.46 a: Down	r=0.00 V=0.51 a: Down	r=0.00 V=0.57 a: Down	r=0.00 V=0.61 a: Down	r=0.00 V=0.54 a: Left	r=-10.00 V=0.00 a: Left	r=-10.00 V=0.00 a: Left	r=-10.00 V=0.00 a: Left	r=-10.00 V=0.00 a: Left	r=-10.00 V=0.00 a: Left	r=-10.00 V=0.00 a: Left	r=0.00 V=6.95 a: Right	r=0.00 V=7.80 a: Right	r=0.00 V=8.76 a: Right	r=0.00 V=9.84 a: Up
r=0.00 V=0.52 a: Down	r=0.00 V=0.57 a: Down	r=0.00 V=0.64 a: Down	r=0.00 V=0.69 a: Down	r=0.00 V=0.63 a: Down	r=0.00 V=0.56 a: Left	r=0.00 V=-0.64 a: Left	r=0.00 V=0.40 a: Right	r=0.00 V=1.72 a: Right	r=0.00 V=3.32 a: Right	r=0.00 V=5.59 a: Right	r=0.00 V=6.19 a: Up	r=0.00 V=6.95 a: Up	r=0.00 V=7.80 a: Up	r=0.00 V=8.71 a: Up
r=0.00 V=0.58 a: Down	r=0.00 V=0.64 a: Down	r=0.00 V=0.72 a: Down	r=0.00 V=0.78 a: Down	r=0.00 V=0.71 a: Down	r=0.00 V=0.64 a: Left	r=0.00 V=-0.60 a: Left	r=0.00 V=0.24 a: Right	r=0.00 V=1.48 a: Right	r=0.00 V=2.96 a: Right	r=0.00 V=4.98 a: Right	r=0.00 V=5.52 a: Up	r=0.00 V=6.19 a: Up	r=0.00 V=6.95 a: Up	r=0.00 V=7.72 a: Up
r=0.00 V=0.65 a: Down	r=0.00 V=0.72 a: Down	r=0.00 V=0.81 a: Down	r=0.00 V=0.89 a: Down	r=0.00 V=0.79 a: Left	r=-10.00 V=0.00 a: Left	r=-10.00 V=0.00 a: Left	r=-10.00 V=0.00 a: Left	r=-10.00 V=0.00 a: Left	r=-10.00 V=0.00 a: Left	r=-10.00 V=0.00 a: Left	r=0.00 V=4.91 a: Right	r=0.00 V=5.51 a: Up	r=0.00 V=6.18 a: Up	r=0.00 V=6.84 a: Up
r=0.00 V=0.72 a: Down	r=0.00 V=0.81 a: Right	r=0.00 V=0.91 a: Down	r=0.00 V=1.01 a: Down	r=0.00 V=0.91 a: Left	r=-10.00 V=0.00 a: Left	r=-10.00 V=0.00 a: Left	r=-10.00 V=0.00 a: Left	r=-10.00 V=0.00 a: Left	r=-10.00 V=0.00 a: Left	r=-10.00 V=0.00 a: Left	r=0.00 V=4.37 a: Right	r=0.00 V=4.91 a: Up	r=0.00 V=5.50 a: Up	r=0.00 V=6.06 a: Up
r=0.00 V=0.81 a: Right	r=0.00 V=0.91 a: Right	r=0.00 V=1.02 a: Right	r=0.00 V=1.14 a: Right	r=0.00 V=1.29 a: Down	r=0.00 V=1.45 a: Down	r=0.00 V=1.65 a: Down	r=0.00 V=1.87 a: Down	r=0.00 V=2.12 a: Down	r=0.00 V=2.40 a: Down	r=0.00 V=2.79 a: Right	r=0.00 V=3.89 a: Right	r=0.00 V=4.37 a: Up	r=0.00 V=4.89 a: Up	r=0.00 V=5.37 a: Up
r=0.00 V=0.88 a: Right	r=0.00 V=1.00 a: Right	r=0.00 V=1.13 a: Right	r=0.00 V=1.27 a: Right	r=0.00 V=1.44 a: Right	r=0.00 V=1.63 a: Right	r=0.00 V=1.85 a: Right	r=0.00 V=2.10 a: Right	r=0.00 V=2.38 a: Right	r=0.00 V=2.70 a: Right	r=0.00 V=3.06 a: Right	r=0.00 V=3.47 a: Right	r=0.00 V=3.89 a: Up	r=0.00 V=4.35 a: Up	r=0.00 V=4.76 a: Up
r=0.00 V=0.85 a: Right	r=0.00 V=0.96 a: Right	r=0.00 V=1.08 a: Right	r=0.00 V=1.21 a: Right	r=0.00 V=1.37 a: Right	r=0.00 V=1.54 a: Right	r=0.00 V=1.73 a: Right	r=0.00 V=1.94 a: Right	r=0.00 V=2.18 a: Right	r=0.00 V=2.45 a: Right	r=0.00 V=2.75 a: Right	r=0.00 V=3.09 a: Up	r=0.00 V=3.47 a: Up	r=0.00 V=3.87 a: Up	r=0.00 V=4.22 a: Up
r=0.00 V=0.80 a: Right	r=0.00 V=0.90 a: Right	r=0.00 V=1.01 a: Right	r=0.00 V=1.13 a: Right	r=0.00 V=1.26 a: Right	r=0.00 V=1.42 a: Right	r=0.00 V=1.58 a: Right	r=0.00 V=1.77 a: Right	r=0.00 V=1.98 a: Right	r=0.00 V=2.21 a: Right	r=0.00 V=2.47 a: Right	r=0.00 V=2.76 a: Right	r=0.00 V=3.09 a: Up	r=0.00 V=3.44 a: Up	r=0.00 V=3.74 a: Up

Figure 3: Question 3 - Value iteration, Action and reward with penalty of -10 for rocks

6 Question 4 - What happens if you run the algorithm for a smaller amount of iterations

With the smaller number of iterations the agent still can find the way to harbour, however this works with some number close to 100 iterations. Figure 4 shows that the agent cannot find the target with too small number of iterations (10 in this case).

As the number of iterations go to infinity, the value function and policy converge to optimal V^* and π^*.
Both, value function and policy need the same amount of iterations to converge, however, I will say that policy requires a bit more iterations that value function.

From the following formula we compute value function as max of expected reward for all actions. $V_{k+1}(s) = max_a E[R_{t+1} + \gamma * V_k(s_{t+1}|s_t = s, a_t = a]$ however, when we compute the policy we take the index from the maximum of the expected future rewards.
$\pi(s) = argmax_a E[R_{t+1} + \gamma * V_k(s_{t+1}|s_t = s, a_t = a]$, now imagine the case when there are more than one state with the same value function, in that case we will have more than one index of the max, and argmax will take one of them (wich may be the wrong), thats why policy need a bit more iterations.

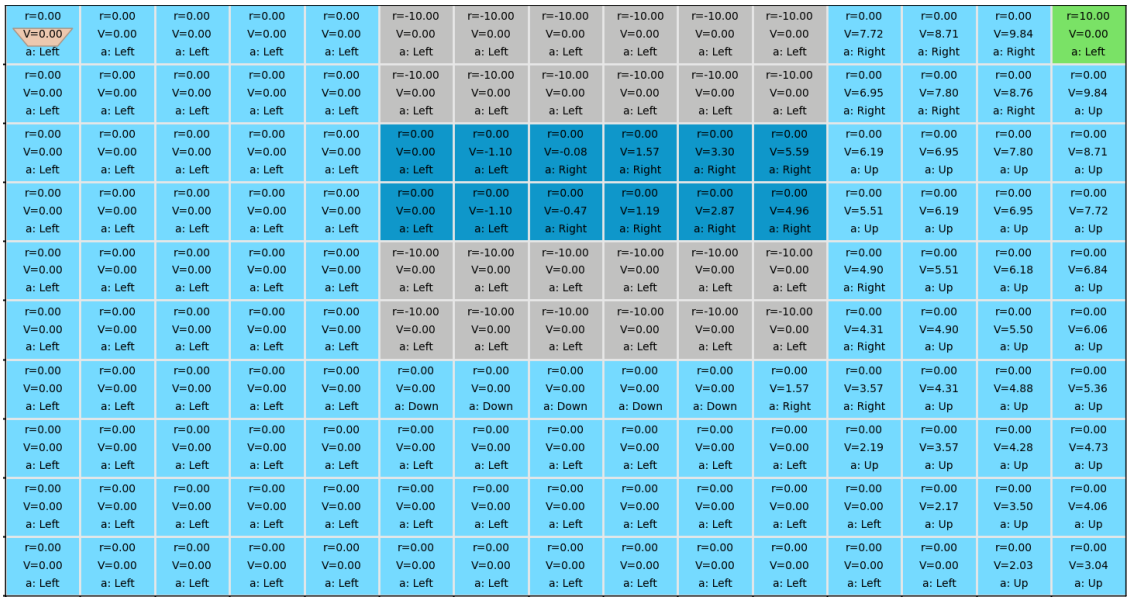

Figure 4: Question 4 - Value iteration, Action - Train for only 10 iterations

7 Task 3

```
#here is the updated function from the first task
def Task1(value_est,policy):
    prev_value_est = value_est.copy()
    for iter in range(100):
        env.clear_text()
        #get all states
        for i in range(env.w):
            for j in range(env.h):
                history = []
                #compute the sum for all s'
                for tran in env.transitions[i,j]:
                    sum = 0.
                    for s_,r,done,p in tran:
                        sum += p * (r + (gamma * value_est[s_
                            [0],s_[1]] if not done else 0))

                    history.append(sum)

                #update V(s), and pi(s)
                value_est[i,j] = np.max(history)
                policy[i,j] = np.argmax(history)

        if ((value_est - prev_value_est) < epsilon).all():
            print('Early stopping')
            break
        else:
            prev_value_est = value_est.copy()

    return value_est,policy
```

Result for task 3 is presented in Figure 5

8 Task 4

$G_t = R_{t+1} + \gamma * R_{t+2} + \gamma^2 * T_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k * R_{t+k+1}$

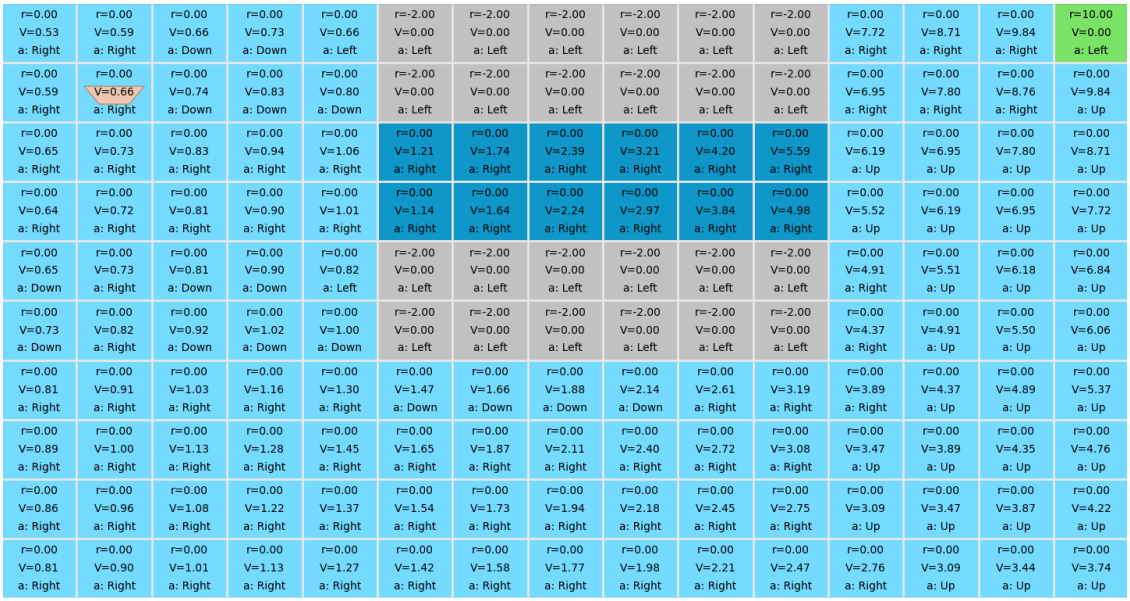

Figure 5: Task 3 - Run until convergence

```
history_G = []
for episod in range(1000):
    state = env.reset()

    done = False
    G, k = 0,0
    while not done:
        action = policy[state]
        state, reward, done, _ = env.step(action)

        G += (gamma**k)*reward
        k+=1
        #env.render()
        #sleep(0.1)

    history_G.append(G)
print('Avg:{}, Std:{}'.format(np.average(history_G), np.std(
   history_G)))
```

I got Avg:0.73, Std:1.35

9 Question 5 - What is the relationship between the discounted return and the value function?

$v_\pi(s) = E[G_t|s_t = s]$ Value function is the expected discounted return.

10 Question 6

If the environment is unknown, we cannot apply value iteration approach presented above. Because, we don't know the motion model $p(s', r|s, a)$, also, the state space may be infinite, or continuous. Using the value function in the real world scenario may be very computational expensive.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.