# Exercise 2

Jari Mattila - 35260T
ELEC-E8125 - Reinforcement Learning

October 10, 2021

## Question 1

What is the agent and the environment in this setup?

The environment is described in this setup as a stochastic process with a probability density that is implemented in practice using state transitions and the associated probabilities and rewards (obtained via env.transitions). The agent is the learner and decision maker who uses information from the environment (i.e. env.transitions) to make a decision about the next action.

## Task 1

Value iteration is implemented as a function in sailing.py. The embedded code snippet is shown below.

Source files: main_Task2.py, sailing.py (function: value_iteration_task1(..)), values_Task1.npy

## Question 2

What is the state value of the harbour and rock states? Why?

The state value of the harbour and rock states is 0.0. When the sailor hits the harbour or rocks, the episode ends. Thus, no transitions from the harbour or rock to the next states are defined (i.e., the probabilities of the next states are all zeros), so that the initial state values (=0) persist across all iterations.

## Task 2

Source files: main_Task2.py, sailing.py (function: value_iteration_task1(..)), policy_Task2.npy

```
def value_iteration_task1(self, value_est, policy, niters=100, gamma=0.9):
    for _ in np.arange(niters):
        for x in np.arange(self.w):
            for y in np.arange(self.h):
                action_values = np.zeros(self.NO_ACTIONS)
                for a in np.arange(self.NO_ACTIONS):
                    state_trans_list = self.transitions[x, y, a]
                    next_value = 0.0
                    for trans in state_trans_list:
                        state, reward, done, prob = trans
                        if state: # if not Null
                            # value iteration: equation (4.10) in Section 4.4 of [1]
                            next_value = next_value + prob * (reward + gamma * value_est[state[0], state[1]])

                    action_values[a] = next_value

                max_action = np.argmax(action_values)

                value_est[x, y] = action_values[max_action]
                policy[x, y] = max_action

    return value_est, policy
```

# Question 3

Which path did the sailor choose, the safe path below the rocks, or the dangerous path between the rocks? If you change the reward for hitting the rocks to -10 (that is, make the sailor value life more), does he still choose the same path?

The sailor chose the more dangerous path between the rocks - and some of the attempts failed badly. If you change the reward for hitting the rocks to -10, the safe path is chosen below the rocks and without crashes.

# Question 4

What happens if you run the algorithm for a smaller amount of iterations? Do the value function and policy still converge? Which of them - the policy or value function - needs less iterations to converge, if any? Justify your answer.

The value function requires approximately 30 iterations for convergence and policy around 24 iterations, assuming $\epsilon = 10e^{-4}$. If the number of iterations is smaller than these numbers, convergence is not guaranteed.

The policy requires fewer iterations to converge. The policy function does not need the exact and final state values: the correct relative of state values of neighbors is enough to have the correct optimal policy (see [Slack]).

# Task 3

Source files: main_Task3.py, sailing.py (function: value_iteration_task3(..)), values_Task3.npy, policy_Task3.npy

# Task 4

The average is $\approx -0.15$ and standard deviation $\approx 0.26$ over $N = 1000$ episodes for the discounted return.

This is a fairly expected result because the random walks have almost always led to rocks (with reward of -2) which explains the negative average value.

Source files: main_Task4.py

# Question 5

What is the relationship between the discounted return and the value function? Explain briefly.

They basically represent the same thing. By definition the value function is the expected value of the discounted reward, i.e. $V(s) = E[G_t|s_t = s]$ where $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$.

In Task 4 if you change the random walks to the optimal policy of Tasks 1-3, the average is around 0.55 and the standard deviation 1.0 because the optimal policy most often ended with a reward of 10 in terminal state (= harbour).

# Question 6

Imagine a reinforcement learning problem involving a robot exploring an unknown environment. Could the value iteration approach used in this exercise be applied directly to that problem? Why/why not? Which assumptions are unrealistic, if any?

The value iteration is used within the Markov decision process (MDP) with the known dynamics (=transition probabilities between states/actions) and reward function, i.e. the environment is known. The robot that explores an unknown environment does not know the Markovian dynamics and the reward function is also unknown therefore the robot has to learn them through exploration.

# References

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