Reinforcement Learning in Ocean Navigation: Solving sea grid-based enviroments with aleatory currents, obstacles and vortices using Q-Learning

Eduardo Manfrellotti
Universita' degli Studi di Milano - Bicocca
MSc Artificial Intelligence for Science and Technology
ID: 921834
E-mail: e.manfrellotti@campus.unimib.it

1. Introduction

This project focuses on training an agent using the Q-learning algorithm to navigate a custom-built grid-world environment. The agent's task is to reach a predefined goal while avoiding obstacles and being affected by vortices that modify its movement probabilities. The environments vary in complexity, from basic configurations with static elements to more challenging setups with randomized and dynamic obstacles. By using Q-learning, the agent iteratively improves its strategy to maximize cumulative rewards, number of average successes and minimizing the number of steps taken to solve the environment. This approach demonstrates the practical applications and effectiveness of RL in solving navigation problems.

2. Definition of the problem

In this project, we tackle *"The Problem of the Storm-Tossed Lighthouse"*, a navigational challenge set in a simulated stormy sea environment. In this problem, the agent, represented by a ship, faces the challenge of navigating through a grid-based sea environment to reach the goal, represented by a lighthouse. The environment simulates a complex stormy sea full of obstacles that can strongly influence the actions taken by the agent. The environment's complexity is increased by the inclusion of obstacles, such as rocks, which can be either fixed or randomized. These obstacles represent hazards like underwater rocks that the ship must avoid to prevent collisions. Additionally, the dynamic currents, along with vortices, add another layer of difficulty. These vortices simulate strong sea currents that influence the ships movement, making navigation unpredictable. The currents can alter the ship's trajectory in unexpected ways, adding to the challenge of steering the ship towards its destination in between the storm.

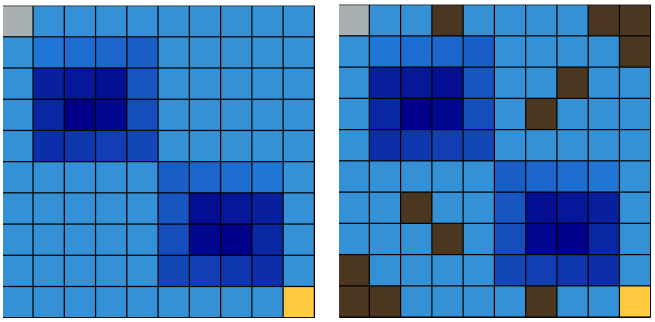
3. Building the environment

To address the problem outlined in Section 2, i have developed a custom grid environment based on the Gym framework. The following paragraphs provide an in-depth overview of the key components and features of this environment.

3.1. Environment Layout

The environment is structured as a 10x10 grid, where each cell represents a segment of the maritime area and can exhibit different characteristics based on the provided configuration options.

The environment offers two major scenarios to vary the difficulty of the navigation task:



(a) Easy environment (b) Hard environment

Figure 1: *Principal possible configurations of the environment*

- Easy Environment (Figure 1a): This is the basic configuration of the environment, designed to provide a less challenging navigation experience for the agent. In this setup, the grid is free of obstacles, allowing the agent to focus entirely on navigating through vortices and sea currents. By not including obstacles, the environment poses a more direct and manageable path, facilitating the agent's learning process. This configuration is specifically designed to help the agent develop fundamental navigation skills and adaptation strategies, without the complexity of additional barriers.

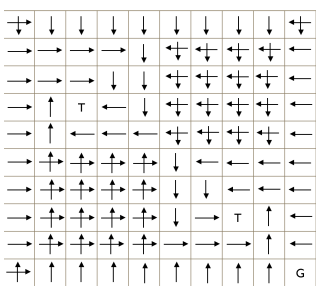
- Hard Environment (Figure 1b): This configuration increases complexity by densely populating the grid with obstacles, represented by rocks. The agent must account for these obstacles and adjust its path accordingly to avoid collisions. The presence of rocks adds an extra layer of complexity to the navigation task, requiring the agent to consider their locations and navigate around them strategically. Additionally, the agent needs to plan and execute movements while avoiding obstacles and adjusting for the effects of currents and sea vortices, thereby raising the overall difficulty of the environment.

3.2. Currents and Vortices

In the described environment, currents and vortices represent the main challenges for the agent. These elements introduce dynamic and unpredictable forces with which the agent must contend while navigating through the grid.

Vortices are the most significant and complex components of the environment. Each vortex is characterized by a fixed dimension of 4x4, and it includes rotation direction, probability, and color grids. The direction and probability grids are closely related: the probability grid indicates the likelihood of the agent being affected by the vortex's current in each tile, while the direction grid shows the direction the agent would move if caught by the vortex. The probability of being trapped by the vortex starts at 5% for the outermost tile and linearly increases to 95% for the second-to-last tile, with the center tile having a 0% probability. These grids are filled based on the vortex's rotation, which can be either clockwise or counterclockwise.

The color grid uses a blue color scale to represent the depth of the vortex. Darker colors indicate greater depth and a stronger pull towards the center. Entering the center of a vortex results in the agent's defeat and the failure of the task, requiring the agent to carefully navigate around these hazards.



0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.05	0.05	0.114	0.178	0.242	0.05	0.05	0.05	0.05	0.05
0.05	0.757	0.821	0.885	0.307	0.05	0.05	0.05	0.05	0.05
0.05	0.692	0.0	0.95	0.371	0.05	0.05	0.05	0.05	0.05
0.05	0.628	0.564	0.5	0.435	0.05	0.05	0.05	0.05	0.05
0.05	0.05	0.05	0.05	0.05	0.242	0.178	0.114	0.05	0.05
0.05	0.05	0.05	0.05	0.05	0.307	0.885	0.821	0.757	0.05
0.05	0.05	0.05	0.05	0.05	0.371	0.95	0.0	0.692	0.05
0.05	0.05	0.05	0.05	0.05	0.435	0.5	0.564	0.628	0.05
0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05

(a) Direction grid (b) Probability grid

Figure 2: *Grids describing the environment*

In the same way, the entire environment, including the vortices, is described by direction and probability grids.

The direction grid for the currents, as shown in Figure 2a, pulls the agent towards the center of the grid. Corners of the grid feature cells with multiple directions that guide the agent towards one of the borders. Vortices are strategically placed in two central areas of the grid, creating regions with higher difficulty. These areas have adjacent cells with dual directions that lead the agent towards one of the vortices, depending on which current affects the agent.

The probability grid for the currents, depicted in Figure 2b, mirrors the arrangement of the vortex probability grids. The rest of the grid simulates an attraction current with a probability equal to that of the outermost vortex tile, maintaining a consistent challenge throughout the environment.

3.3. Obstacles

Obstacles in the environment, represented by rocks, add another layer of difficulty to the agent's navigation task. These obstacles are strategically placed on the grid, creating barriers that the agent must navigate around.

In the basic configuration (Figure 1a), there are no obstacles, allowing the agent to focus on learning to navigate the currents and vortices. However, beginning from the hard configuration (Figure 1b), rocks are introduced, requiring the agent to consider their positions and adjust its path accordingly to avoid collisions.

The rocks are static and inaccessible, which means that the agent cannot move through these cells. If the agent collides with a rock, he must alter his course to find an alternative route. As we will see in the next section, the placement of some rocks may be randomised in each episode, so that the agent cannot simply memorise routes and must instead develop a robust strategy to avoid obstacles.

3.4. Randomness

Randomness is introduced as an additional element of difficulty in the designed environment, enriching the overall challenge and complexity faced by the agent. By incorporating various forms of randomness, the environment ensures that each episode offers a unique and unpredictable scenario.

One primary source of randomness lies in the initial placement of both the agent and the goal. Depending on the configuration settings, these positions can either be fixed or randomly determined within specific constraints. Specifically, the agent can only be placed in the first column of the grid, while the goal is positioned in the last column. If rocks are present in the environment, both the agent and goal are placed in positions that avoid these rock locations. This variability requires the agent to adjust its strategy for each new episode, as the starting and goal locations may differ every time.

Additionally, the distribution of some rocks can also be randomized. When rocks are included, their positions can follow specific patterns or be chosen randomly from a set of predefined patterns. This introduces further variability in the environment, compelling the agent to navigate around obstacles that may shift in each episode.

Finally, the randomness extends to the influence of water currents on the agent's movement. Each cell in the grid is associated with a probability that determines whether the

agent's intended action will be overridden by the direction of the current. As a result, even when the agent selects a particular action, there is a chance it will be redirected by the current, adding an element of unpredictability to its movement. This requires the agent to continuously adapt to potential deviations caused by the currents.

4. Q-Learning

To accomplish the task described in Section 2, the Q-Learning algorithm was employed. Q-Learning is a model-free reinforcement learning algorithm used to estimate the value of taking a certain action in a given state. As a model-free method, it does not require knowledge of the environment's dynamics, making it well-suited for problems with stochastic transitions and rewards. The primary objective of Q-Learning is to approximate the optimal action-value function, which is derived from the Bellman equations.

4.1. Algorithm Description

- **Initialization:** The Q-table, which stores the expected future rewards for each state-action pair, is initialized to zeros. In the environment considered, states are defined by the positions of the agent and the goal, while actions correspond to the possible moves of the agent.

- **Action Selection:** The agent uses an epsilon-greedy strategy to balance exploration and exploitation. With probability ϵ, the agent explores the environment by selecting a random action. With probability $(1 - \epsilon)$, the agent exploits its current knowledge by choosing the action with the highest Q-value for the current state.

- **Updating Q-Values:** After taking an action, the agent receives a reward and transitions to a new state. The Q-value for the state-action pair is updated using the following Q-Learning update rule, which is based on the Bellman equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \quad (1)$$

  Here, α is the learning rate, r is the immediate reward, γ is the discount factor, s' represents the new state, and a' denotes possible actions in the new state. This update rule is grounded in the Bellman equation for the optimal action-value function:

$$Q(s,a) = \mathbb{E}\left[r + \gamma \max_{a'} Q(s',a') \mid s,a \right] \quad (2)$$

- **Exploration vs. Exploitation:** Over time, ϵ is linearly decayed to gradually shift from exploration to exploitation, allowing the agent to refine its policy based on accumulated experience.

In the context of this project, Q-Learning is adapted to handle various configurations of the grid environment, ranging from simple setups to more complex scenarios involving obstacles and randomization. The Q-table is customized to reflect the grid size and specific challenges of the environment, ensuring that the agent learns an optimal policy tailored to each scenario.

5. Training setup

To implement the Q-Learning algorithm, a custom training setup was implemented, analyzing a variety of environment configurations, each with an associated number of training episodes. The training setup consists of five distinct environment configurations:

- **easy_env**: basic configuration with a static agent and goal
- **"r_easy_env"**: basic configuration with randomized positions for the agent and goal
- **"hard_env"**: basic configuration with static rocks as obstacles
- **"r_hard_env"**: hard configuration with some rocks randomized following two predefined patterns
- **"rr_hard_env"**: randomized hard configuration with both randomized rocks and positions for the agent and goal

These configurations and their specific parameters are detailed in Table 1.

Configuration	Max Ep. Steps	Training Episodes
easy_env	150	20000
r_easy_env	150	20000
hard_env	300	20000
r_hard_env	450	50000
rr_hard_env	450	40000

TABLE 1: *Configuration details for Q-Learning training setup*

For each configuration, the Q-Learning algorithm was executed with a maximum number of steps per episode (max_ep_steps) and a predefined number of training episodes.

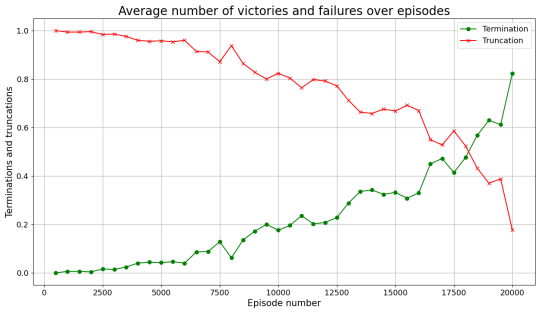
The Q-Learning parameters were set as follows: the learning rate (α) was set to 0.9, the discount factor (γ) was set to 0.9, and the exploration rate (ϵ) was initialized at 1, gradually decaying over time to balance exploration and exploitation.

The training process involved saving the Q-table model and statistics at regular intervals (every 500 episodes), ensuring that the agent's performance could be analyzed and refined across different environment setups.

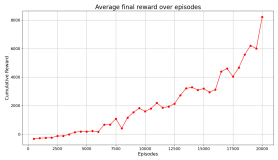
6. Results

The performance of the Q-Learning algorithm was evaluated across all the different environment configurations. The results are illustrated through three key metrics for each configuration: average episode length, average cumulative reward, and average terminations and truncations. Below are presented the resulting plots for each configuration.

6.1. Easy Environment



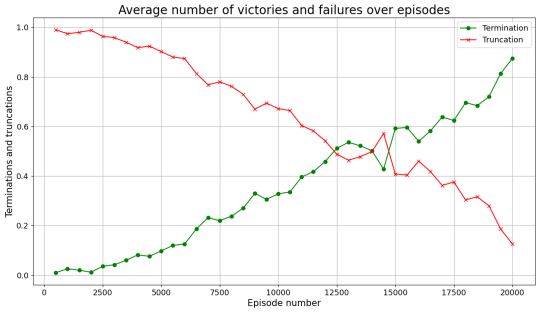
(a) Average terminations and truncations



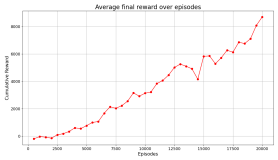
(b) Average episode length (c) Average cumulative reward

Figure 3: *Results for the Easy Environment configuration*

6.2. Randomized Easy Environment



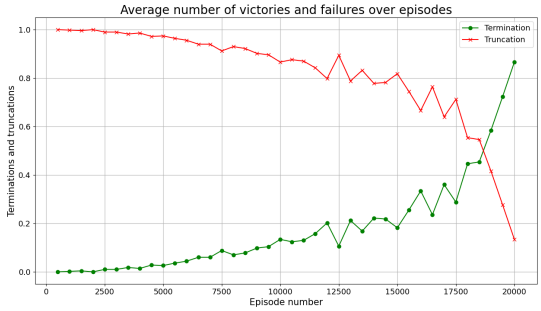
(a) Average terminations and truncations



(b) Average episode length (c) Average cumulative reward

Figure 4: *Results for the Randomized Easy Environment configuration*

6.3. Hard Environment

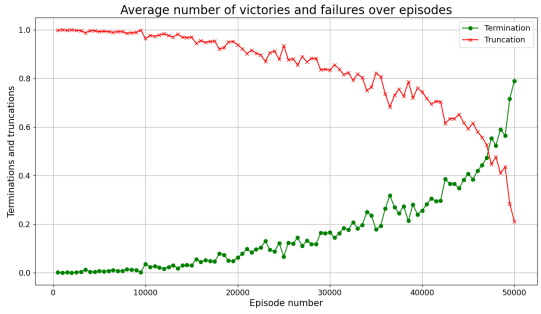


(a) Average terminations and truncations

(b) Average episode length (c) Average cumulative reward

Figure 5: *Results for the Hard Environment configuration*

6.4. Randomized Hard Environment



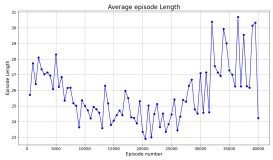
(a) Average terminations and truncations

(b) Average episode length (c) Average cumulative reward

Figure 6: *Results for the Randomized Hard Environment configuration*

6.5. Randomized Hard Environment with Random Agent and Goal

(a) Average terminations and truncations



(b) Average episode length

(c) Average cumulative reward

Figure 7: *Results for the Randomized Hard Environment with Random Agent and Goal configuration*

7. Comments

As it can be evaluated from the resulting tables, the results obtained from Q-Learning across all tested environments demonstrate overall strong performance. All environments are successfully solved, with success rates ranging between 80% and 90%. The average final reward achieved in each environment falls between 8000 and 9000, indicating the algorithms robust capability to optimize learning strategies. Additionally, the number of moves required to achieve these results ranges between 20 and 30, reflecting the capacity of the model to identify an efficient path across different environments.

The variable number of training episodes and maximum steps per episode was chosen based on extensive testing. These parameters were adjusted to balance between computational efficiency and the quality of learning. For simpler environments, fewer episodes and shorter maximum steps were sufficient to reach optimal performance, while more complex environments required increased episodes and longer maximum steps to ensure adequate exploration and learning. This iterative tuning process ensured that the Q-Learning algorithm could effectively handle different environment complexities.

These outcomes highlight not only the robustness of Q-Learning in handling environments of varying difficulty but also its ability to adapt and optimize performance across diverse learning scenarios.

8. Possible improvements

Even if the Q-Learning algorithm has demonstrated really good performance, there are several ways for potential improvement.

One key area is the exploration of alternative reinforcement learning algorithms that may offer enhanced capabilities or efficiencies. For instance, algorithms such as Deep Q-Networks (DQN), Proximal Policy Optimization (PPO), or Soft Actor-Critic (SAC) might provide better performance through advanced techniques like experience replay, policy optimization, or entropy regularization, respectively. These methods can address issues related to exploration and exploitation, potentially leading to more robust and efficient learning.

Another possible improvement is hyperparameter tuning. Fine-tuning parameters such as learning rates, discount factors, and exploration strategies can significantly impact the performance of the algorithm. A systematic approach to hyperparameter optimization, using techniques such as grid search or random search could further improve learning outcomes and reduce the time required to achieve optimal performance.

Additionally, employing libraries like Stable-Baselines3 could streamline the experimentation process.