Q-learning Frozen Lake

Arne Lescrauwaet (852617312) [1] **Tobias Leenders (85253398)** [1]

1. Brief Introduction

In our project, we are tasked with learning an agent to traverse a frozen lake without falling into the water. The agent learns by trial-and-error, adjusting the actions it takes based on the rewards it received in the past.

We will use the Q-learning algorithm. This algorithm generates a table called the Q-table which has a mapping of every state and possible action to a value. The agent will learn which actions to take based on the values of this table. The code for this project is available in the Github repo [1]

1. How does the behavior of the agent differ when using a high or low value for the exploration-exploitation (ϵ) parameter
2. Does the discount factor (γ) have a noticeable impact on the score achieved by the agent
3. Does the learning rate (α) have a noticeable impact on the score achieved by the agent

2. Methods

Q-learning is a value-based algorithm wherein a value-function is trained to produce the value associated with a given state-action pair. Subsequently, this value-function guides a preselected policy in making decisions and executing actions. The crux of the methodology lies in the pursuit of an optimal value function, as its attainment inherently leads to an optimal policy. Q-learning achieves this objective through the initialization of a Q-table, serving as a comprehensive repository that maps state-action pairs to their respective Q-values. Action selection is then facilitated by referencing this table, and its refinement occurs through the iterative application of the Bellman equation. The algorithm can be tuned by adjusting its parameters. In this project, we will focus on three parameters: the exploration-exploitation (ϵ) parameter, the discount factor (γ) and the learning rate (α)(Devlieger, 2022; Qle)

The exploration-exploitation parameter dictates how probable the agent is to choose the best action versus trying a new action. The default value of the exploration-exploitation parameter in the starter repository is 0.5 meaning that the algorithm is as likely to explore as it is to exploit since random.random() has a range of 0.1 to 1.0. We will compare the results of the agent using the default parameter versus a low (0.1) and a high (1.0) value. We will also see how a technique called epsilon decay fairs against a static value. Epsilon decay is a technique where the algorithm starts off with a high epsilon value to focus more on exploring and with more iteration the epsilon value slowly decreases to focus more on exploitation as it has filled the q-table more.

The discount factor (γ) influences how much importance is placed on future rewards. The default value is 0.9 which decreases the value of future rewards quite a bit since the range of the discount factor is between 0 and 1. This makes the agent focus more on short-term rewards. Again we will compare the behaviour of the agent/ results using this value versus a more moderate value and a low value

The learning rate (α) indicates how much importance we assign to the last update versus the previous values. The higher the learning rate the quicker a new Q-value is replaced. We are going to compare the results from a high value learning rate with a medium and low value.

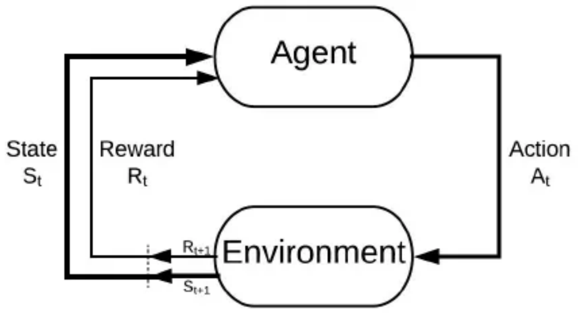


Figure 1. The basic components of reinforcement learning

[1]Link to Github: https://github.com/4rn3/KTA_q_learning

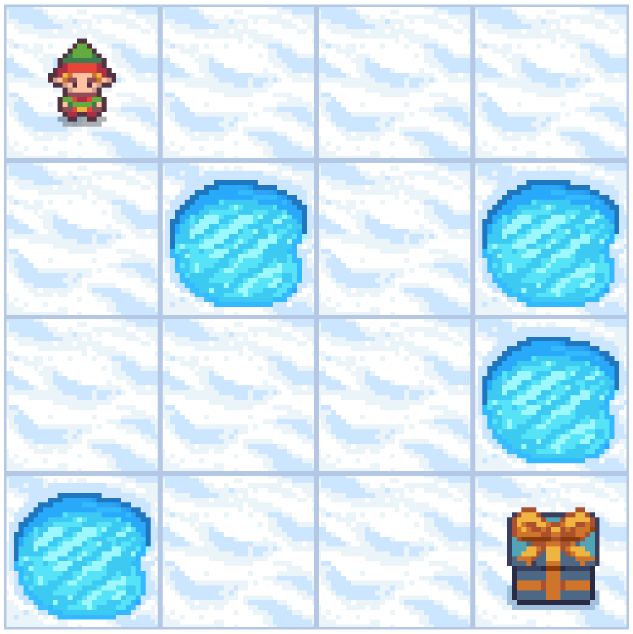


Figure 2. Frozen Lake environment

3. Experimental Results

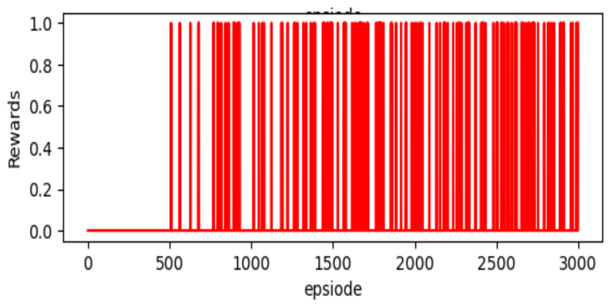
3.1. Environment

Reinforcement learning makes use of an environment instead of datasets. Our environment is the Frozen Lake environment from Gymnasium. In this environment the agent controls a figure and tries to navigate its way across the map avoiding the frozen lakes to arrive at the present. The environment has an observation space that is represented as current row x number of rows x current column. The agent always starts at the same position with coordinates [0,0]. The agent has 4 moves, it can move up, left, down and right. The rewards for the environment are as follows: when the agent reaches its goal it gets one point, if it falls in the ice or fails to reach the present it gets 0 points.(Gym)

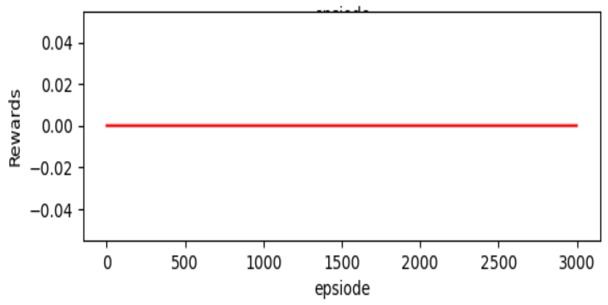
3.2. Results

3.2.1. Hyperparameters

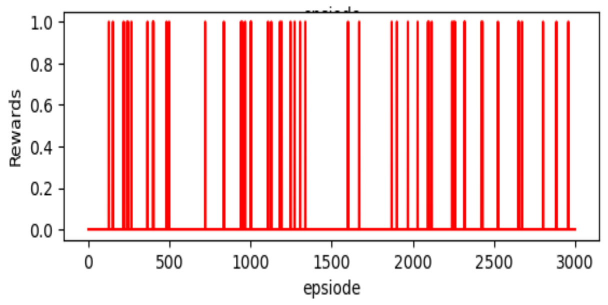
epsilon:
Training the agent with the default epsilon parameter value of 0.5 results in predictable reinforcement learning behavior. We see that the agent fails to complete the level for the first 500 episodes. During this time the agent was filling the q-table. In the next 1000 episodes (500-1500) we see the agent starting to complete the level more often. The last 1500 episodes the agent has become quite capable of clearing the level.



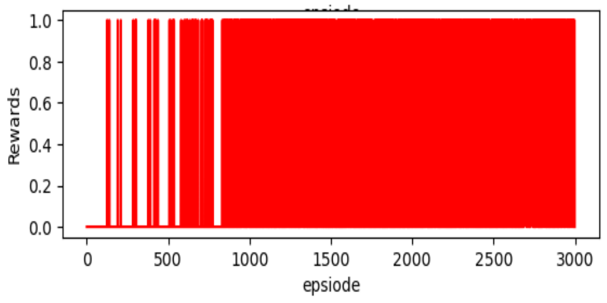
For the low epsilon training we went with a very low value: 0.1 to clearly show the impact the parameter has. When epsilon is low the agent is more likely to exploit known moves rather than explore new moves. With a value of 0.1 the agent practically doesn't explore new moves and this is exactly what we saw in this case. The agent stuck to the one move it had experience with (moving one tile down). This caused the agent to fail to complete the level a single time.



When training the high epsilon agent we went with the other extreme and chose a value of 1. Because of this the agent only explores new moves. Despite not exploiting better moves the agent manages to clear the level a few times purely based on luck.



Finally, we trained the agent using epsilon decay. This resulted in the best results by far. The agent started off exploring and filling the q-table which we can see in the first 500 episodes by the low amount of cleared levels. In the next 500 episodes we already see the exploitation increase, this is seen by the increase in level clears. In the last 2000 episodes the agent primarily focuses on exploiting the gained knowledge and we see this because the agent didn't fail a single level in the period.



Lastly, we also compared the average rewards of the different agent in the following table

Table 1. Epsilon tuning

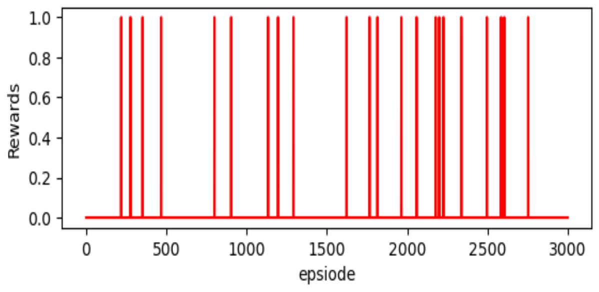
ϵ	Average Reward	Runtime
0.5	0.0563	0.9306
0.1	0.0000	0.3888
1	0.0127	0.6972
1 - 0.1	0.3817	1.966

We also calculated the total rewards after 3000 episodes per configuration:

Table 2. Total rewards after 3k episodes

ϵ	γ	α	Total reward	Completion
0.5	0.9	0.1	132	4.4%
0.1	0.9	0.1	0	0.0%
1	0.9	0.1	33	1.1%
1 - 0.1	0.9	0.1	1110	37.0%

gamma:
To demonstrate the effect of a low γ value, we set it to the minimum value of 0. This means that the agent will not take future rewards into account and will only consider immediate rewards. Our environment only has a single reward if the agent reaches the goal, this means that the agent should perform better with a higher discount factor.



We would also like to demonstrate the effect of a high γ value, so we set it to the maximum value of 1. Now the agent will consider the future rewards equally as important as the immediate reward. This mostly produced good results, but we observed cases where it could get stuck in an infinite loop. In the chosen environment, there are no negative rewards, and the agent only receives a reward of 1 for reaching the end. If the discount factor is set to 1, the agent will have infinite time to complete the task, provided it does not step into one of the holes. However, this can cause the agent to get stuck in an infinite loop of walking back and forth without reaching the goal or falling into one of the holes. To prevent this from happening, it is recommended to use a discount factor below 1.

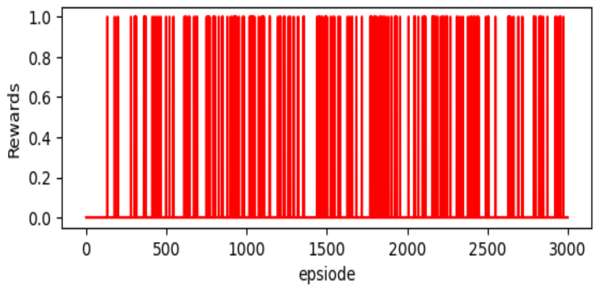
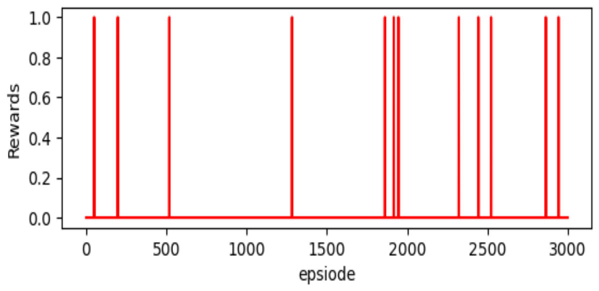


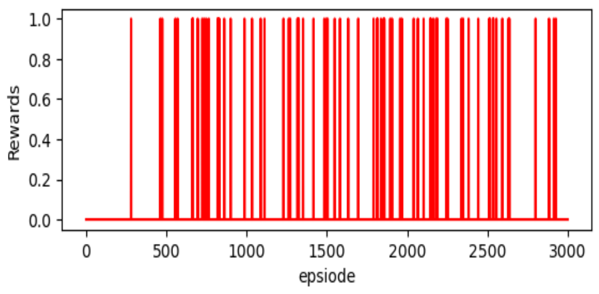
Table 3. Total rewards after 3k episodes

ϵ	γ	α	Total reward	Completion
0.5	0.9	0.1	132	4.4%
0.1	0.9	0.1	0	0.0%
1	0.9	0.1	33	1.1%
1 - 0.1	0.9	0.1	1110	37.0%

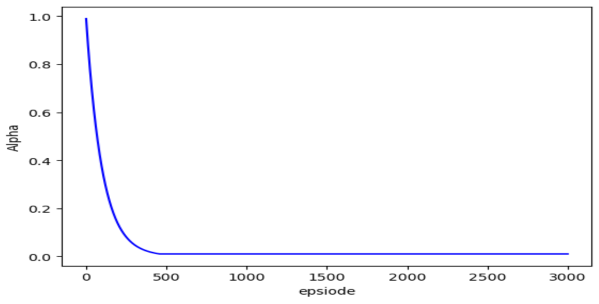
alpha:
The default α value used for training the agent was set at 0.1 which results in the same results we have seen for the default epsilon parameter. The agent will fail to complete the level for around the first 500 episodes. Afterwards we can see the agent starting to complete the level quite often. When we lower the learning rate to 0 we can see that the agent will only complete the level very rarely, seemingly by chance. This is because at 0 the agent will learn nothing.



Setting α to 1 we can see Setting α to 1 we can see that the performance is worse compared to the default value of 0.1. This is because at this value the Q-values are being updated too aggressively which causes it to overshoot optimal Q-values.



Another option that was explored was to use a decaying value for α. Starting out at a value of 0.99 and decaying to a value of 0.01, this prevents the overshooting seen in the constant high learning rate.



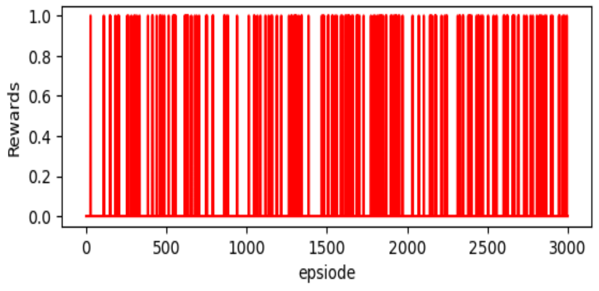
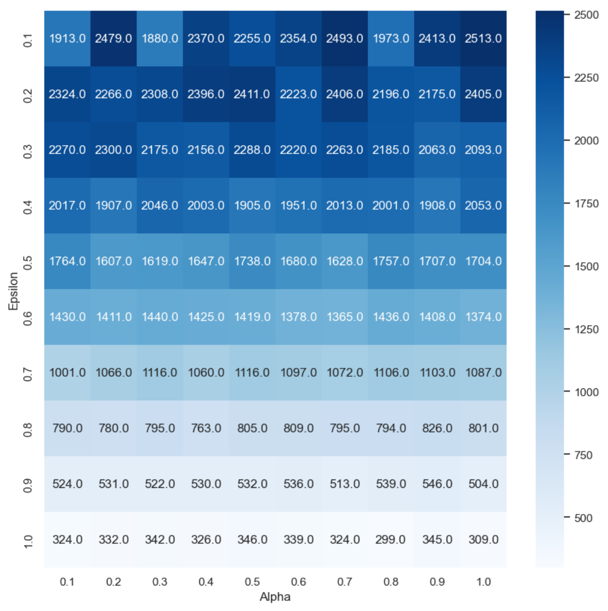


Table 4. Total rewards after 3k episodes

ϵ	γ	α	Total reward	Completion
0.5	0.9	0.1	132	4.4%
0.1	0.9	0.1	0	0.0%
1	0.9	0.1	33	1.1%
1 - 0.1	0.9	0.1	1110	37.0%

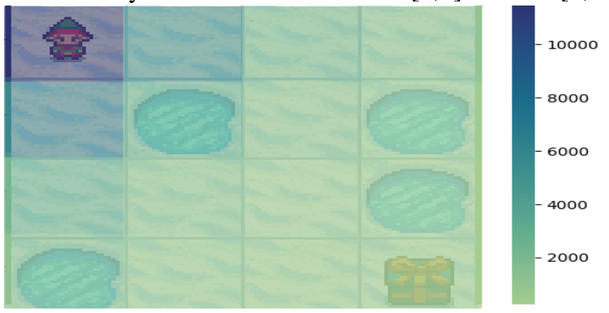
Hyperparameter heatmap:

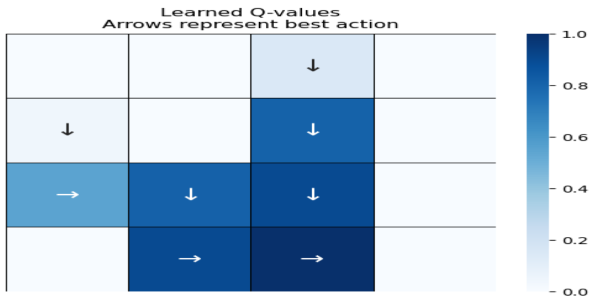


Epsilon \ Alpha	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	1913.0	2479.0	1880.0	2370.0	2255.0	2354.0	2493.0	1973.0	2413.0	2513.0
0.2	2324.0	2266.0	2308.0	2396.0	2411.0	2223.0	2406.0	2196.0	2175.0	2405.0
0.3	2270.0	2300.0	2175.0	2156.0	2288.0	2220.0	2263.0	2185.0	2063.0	2093.0
0.4	2017.0	1907.0	2046.0	2003.0	1905.0	1951.0	2013.0	2001.0	1908.0	2053.0
0.5	1764.0	1607.0	1619.0	1647.0	1738.0	1680.0	1628.0	1757.0	1707.0	1704.0
0.6	1430.0	1411.0	1440.0	1425.0	1419.0	1378.0	1365.0	1436.0	1408.0	1374.0
0.7	1001.0	1066.0	1116.0	1060.0	1116.0	1097.0	1072.0	1106.0	1103.0	1087.0
0.8	790.0	780.0	795.0	763.0	805.0	809.0	795.0	794.0	826.0	801.0
0.9	524.0	531.0	522.0	530.0	532.0	536.0	513.0	539.0	546.0	504.0
1.0	324.0	332.0	342.0	326.0	346.0	339.0	324.0	299.0	345.0	309.0

3.2.2. ENVIRONMENT

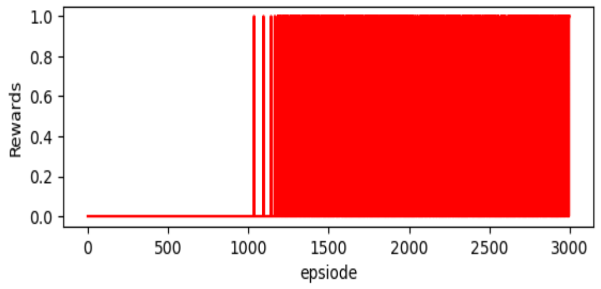
Default parameter heatmap:
We made a heatmap of the tiles traversed by the agent trained on default parameters to visualize the routes it took during the 3000 episodes of training. The results were as to be expected with the most visits on tiles close to the starting point. We see that the two most deadly lakes are the ones at [1,1] and at [0, 3].





Slippery movement:
The frozen lake environment is slippery, meaning that for every move the agent makes there is a 1/3 probability that it moves in either perpendicular direction. The environment is random in nature, with a quantified probability of 2/3 of

moving in the wrong direction and is therefore a stochastic environment. It is possible to disable this by setting `is_slippery` to false, this turns the environment into a deterministic environment. With this property set to false every move made by the agent will be correctly executed meaning and it will be possible to accurately determine the next state of the agent.



Because the environment is now deterministic the agent can learn how to solve it without fail. After the agent has found this solution it will be able to reach the goal very consistently.

Random starting position:
Randomizing the starting position in our default environment:

3.3. Discussion

The environment we used has the option to make your own maps via a grid like structure of letters. It could be interesting to see if the results are further exaggerated or converge when compared on more complex maps. It would be interesting to see what the actual optimal parameters are, this could be found via hyperparameter tuning tools such as Optuna or gridsearch (Opt)

References

Gymnasium gymnasium frozen lake. https://gymnasium.farama.org/environments/toy_text/. Accessed: 2023-10-17.

optuna optimize your optimization. https://optuna.org/. Accessed: 2023-10-11.

Chathurangi Shyalika a beginners guide to q-learning. https://towardsdatascience.com/a-beginners-guide-to-q-learning-c3e2a30a653c. Accessed: 2023-10-11.

Devlieger, I. Les2 value based. 2022.