Tabular RL

DAT440 Advanced topics in machine learning

Group 10

Ludvig Tydén

Jonas Nordin

May 19, 2023

Contents

Introduction

This report is done as part of the course DAT440 Advanced techniques in machine learning at Chalmers University of Technology.

1 Describe the algorithms

In this section the algorithms are described, how they work and what seperates them.

Q-learning

Q-learning is an algorithm that enables an agent to learn an optimal policy by making decisions based on the expected future rewards of its actions. This expected reward is called the Q value and is stored in the Q-table which represents states (rows) and actions (columns).

The algorithm work by updating the Q-value during training by observing the current state of the environment and selecting an action based on an exploration/exploitation strategy. After taking an action the agent receives a reward and updates the Q-value for the state action-pair based on the Bellman equation, shown i equation 1.

$$Q(S,A) = Q(S,A) + \alpha \left[R + \gamma \max_a Q(S',a) - Q(S,A) \right] \tag{1}$$

In equation 1 α represents the learning rate, R is the reward and γ is the discount factor. By iteratively updating the Q-values of the state-action pairs an optimal policy can be learned that maximizes the expected cumulative reward. When found the agent can use the Q-values to select the best action for a given state by choosing the action with the highest Q-value.

Double Q-learning

The Double Q-learning algorithm is an extension of the Q-learning algorithm that uses two separate Q-tables. This is done to avoid overestimation of the expected reward that can occur when using a single Q-table. The first Q-table is used for estimating the value of the selected action and the second Q-table is used for selecting the action.

The double Q-learning algorithm updates the two Q-tables separately using the bellman equation, this is shown in equations 2 & 3.

$$Q_1(S,A) = Q_1(S,A) + \alpha \left[R + \gamma Q_2(S', \arg\max_a Q_1(S',a)) - Q_1(S,A) \right] \tag{2}$$

$$Q_2(S,A) = Q_2(S,A) + \alpha \left[R + \gamma Q_1(S', \arg\max_a Q_2(S', a)) - Q_2(S,A) \right] \quad (3)$$

SARSA

SARSA (State-Action-Reward-State-Action) is an on-policy algorithm, unlike Q-learning which is off-policy. An on-policy learning algorithm learns the value function of the policy that it is following. Firstly, the Q-table is initiated as in the Q-learning algorithm. The Q-values are then updated using equation 4 below.

$$Q(S,A) = Q(S,A) + \alpha \left[R + \gamma Q(S', A') - Q(S,A) \right] \quad (4)$$

The equation depends on both the current state-action pair and the next state-action pair, following a specific policy and not the maximum Q-value as in the Q-learning algorithm. This results in that SARSA learns the value function relating to the used policy.

Expected SARSA

The Expected SARSA algorithm is a modification of the SARSA algorithm. In contrast to SARSA, Expected SARSA estimates the value of the next action by considering the expected value of all possible actions in the next state, weighted by their probabilities of being chosen according to the agent's current policy (in this case epsilon greedy).

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t) \right] \quad (5)$$

The above equation is then used to update the Q-value using the sum value of all possible actions in the next state multiplied with the probabilities of all possible actions in the next state.

2 Run and plot rewards

The moving average of the rewards for each agent in the two environments is shown in Appendix A.

3 Visualize Q-values and greedy policy

The visualizations of the learned policies and Q-values for every combination of agent and environment is shown in Appendix B. Since the nature of each agent is stochastic, each run will produce a different result, meaning that running the same agent in the same environment twice may produce different policies. The

optimal policy would be the policy that the agent learns that results in the most expected cumulative reward over time. As the agent interacts with the environment it updates and improves its policy to achieve a high reward. In the FrozenLake environment, the agents have a risk of "slipping" and choosing either the left, with probability of 1/3, or right action, with probability of 1/3, (relative to the chosen action) or continuing with the chosen action with the probability of 1/3. Hence, the optimal policy for the agent in FrozenLake would be to choose an action were it has no risk of falling into the water.

In the RiverSwim environment the optimal policy for the agent would be to choose to go right at every state, to maximize the reward. All the agents except the SARSA agent learns the optimal policy. Since there is a small reward if the agent chooses to go left in state zero, and if choosing right the agent has a 40% risk of staying in state zero, it may learn to just stay in state zero and collect rewards by going left. Although, this is a sub-optimal policy.

4 Initialization of Q-values

During the implementation of this assignment, the Q-values have been initiated with random values between 0 and 1. Thus, paths are initially valued approximately the same. However, initializing Q-values optimistically (with large random values) makes the agent initially value all paths as great and therefore encourages it to explore more during training. This optimism may help the agent discover rewarding paths faster and receive higher rewards during training compared to initializing with zero or low random values.

Using an optimistic initial Q-table may in the end also affect the final policy learned by the agent. The agent being more biased towards exploration makes it tend to take actions that it might have considered suboptimal initially but can lead to greater rewards. Consequently, the greedy policy obtained from optimistic initialization might exhibit more exploration and exhibit risk-taking behavior to maximize long-term returns.

Conceptually one can think of it as follows. Initializing the Q-table with small values, the agent rewards paths for giving good returns and if initializing it with large values the agent rather penalizes paths depending on how good they are.

In the case of the RiverSwim and FrozenLake environments, which are relatively simple, we will not see any bigger differences in the policies learned by the agents as most of the already finds the optimal paths. One difference we will notice depending on the size of the values in the Q-table is the time it takes for it to converge. This difference is seen in figure 1 and 2 in appexdix A where the latter is initialized with a q-table having values of three orders of magnitude larger than the former.

5 Discussion

As one can see in Figure 1 in Appendix A, the Q-learning algorithm performs well and even better than the double Q-learning algorithm seen in Figure 3. The Q-learning algorithm is off-policy, meaning that the agent can explore the environment more using a different exploration policy, while learning the optimal policy. Unlike on-policy algorithms, where the agent uses the learned policy when choosing actions. Since the FrozenLake environment is simple, the Q-learning agent can learn the optimal policy quickly as seen in Figure 1.

In an environment more complex than FrozenLake, the double Q-learning agent would possibly perform better since it avoids overestimating the Q-values. But in the simplicity of both of the environments the Q-learning agent dose not tend to overestimating. Thus, the double Q-learning agent converges slower to the optimal policy.

The SARSA algorighm is on-policy, meaning that it uses the current policy to choose the action and then tries to improve that policy over the learning session. As seen in the reward plot in Figure 4 in Appendix A and the policy and Q-values for SARSA, shown in Figures 18 & 19, The SARSA agent is the worst performing agent i the FrozenLake environment. It is also the worst performing agent i RiverSwim, seen in Figures 20 & 25. As SARSA follows the current policy for exploration and exploitation it may explore less than, for example, Q-learning which is off-policy and can explore the environment more using ϵ-greedy. As mentioned earlier, the stochastic nature of the environments will result in different results for each run. This is extra clear for the SARSA agent, which can learn the optimal policy in RiverSwim for on run, but next time learn the sub-optimal policy of only choosing to move left. SARSA getting stuck in a "local maximum" can be caused by the values of the hyper-parameters, such as learning rate or ϵ. Changing the learning rate may cause the SARSA agent to avoid this "local maximum" and reach the optimal policy. Changing the value for ϵ can lead ta an increase in exploration, enabling the agent to find the optimal policy.

The modified version of SARSA, expected SARSA, does perform better than the normal version of it, especially in the FrozenLake environment but also in RiverSwim. This is most likely due to the difference in how the models update their Q-values. The key difference is that the modified version takes the probabilities of the next possible actions into account when updating a Q-value, the normal version instead bases its updates on an actual action taken rather than probabilities of the possible actions being taken. This allows the expected SARSA model to have a more informed exploration strategy than the normal version, which may be more prone to being stuck in sub-optimal paths because of the less informed updates of Q. Expected SARSA also tends to perform better than SARSA in stochastic environments which is clearly noticeable in figures 4, 8, 5 and 9 in A.

Appendices

A Reward plots

A.1 FrozenLake

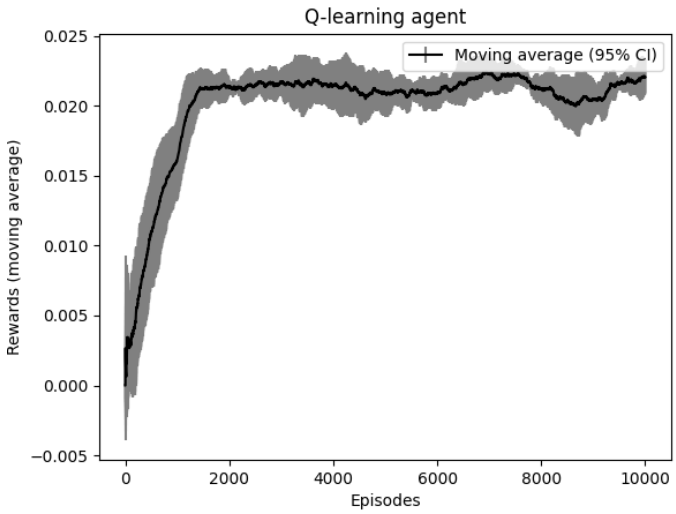

Figure 1: Average moving reward for Q learning in FrozenLake environment

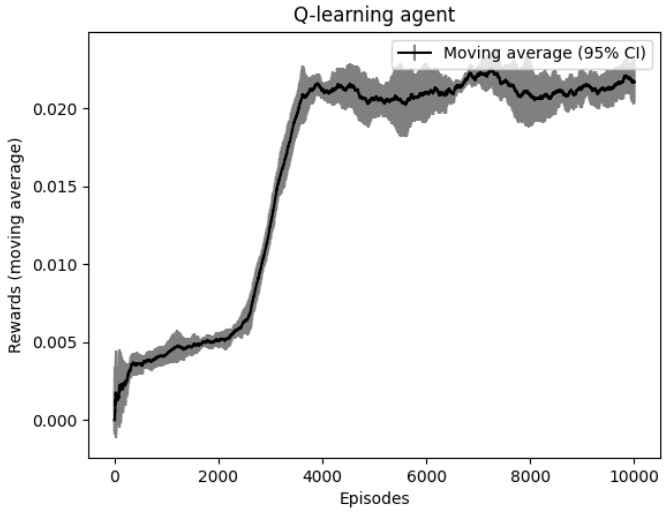

Figure 2: Average moving reward for Q learning in FrozenLake environment with an optimistically initialized Q-table

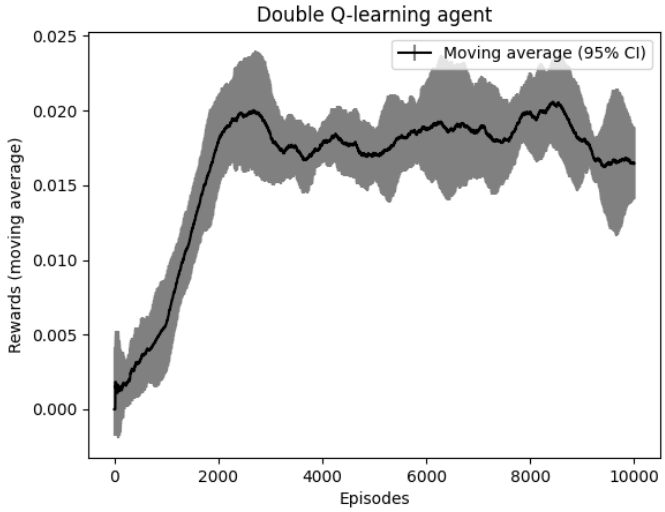

Figure 3: Average moving reward for double Q learning in FrozenLake environment

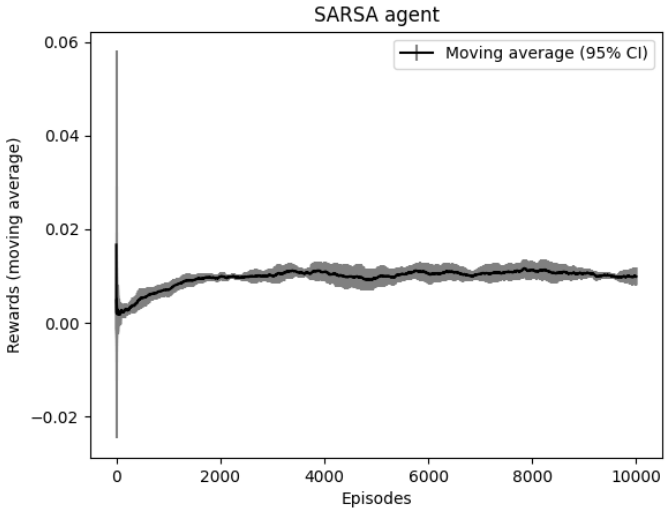

Figure 4: Average moving reward for SARSA in FrozenLake environment

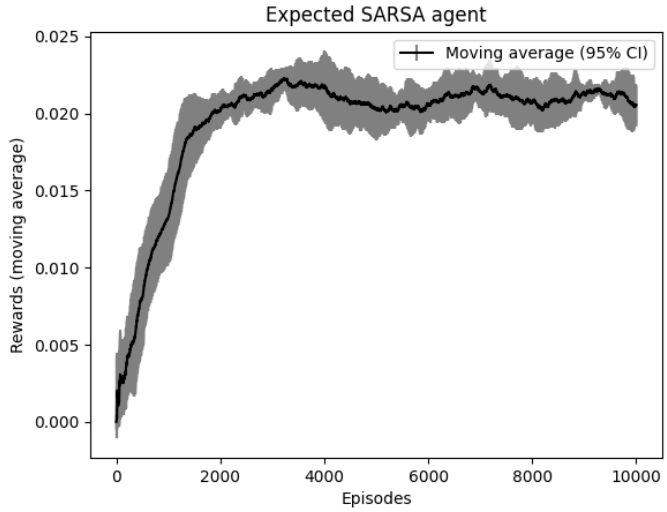

Figure 5: Average moving reward for expected SARSA in FrozenLake environment

A.2 RiverSwim

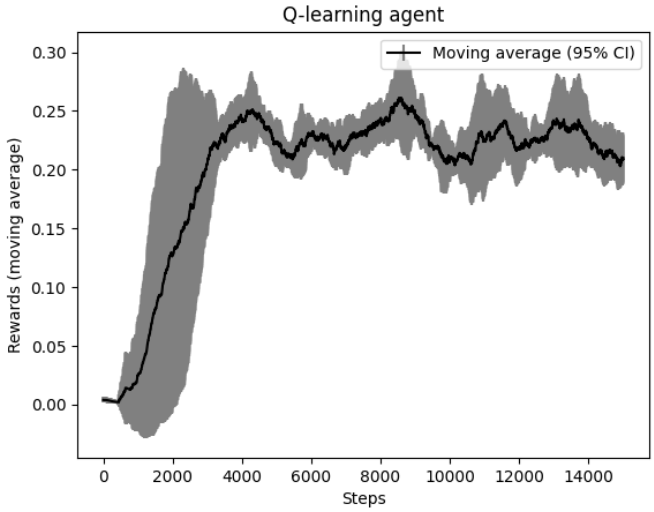

Figure 6: Average moving reward for Q learning in RiverSwim environment

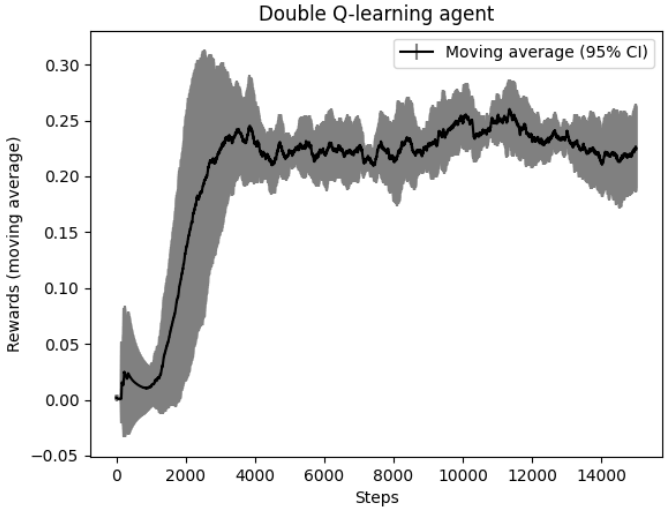

Figure 7: Average moving reward for double Q learning in RiverSwim environment

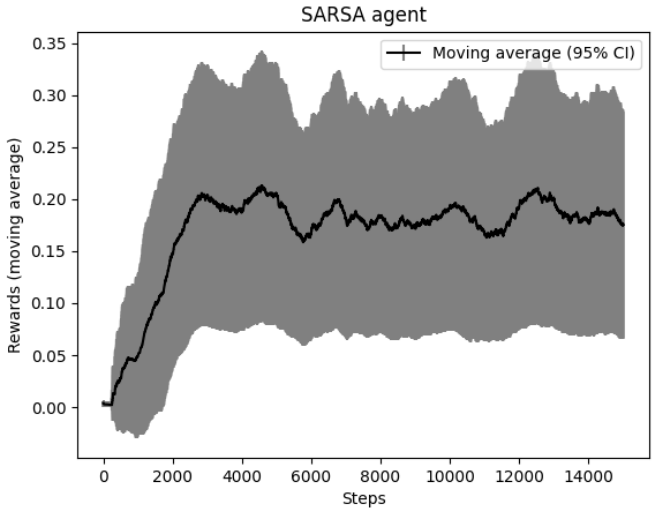

Figure 8: Average moving reward for SARSA in RiverSwim environment

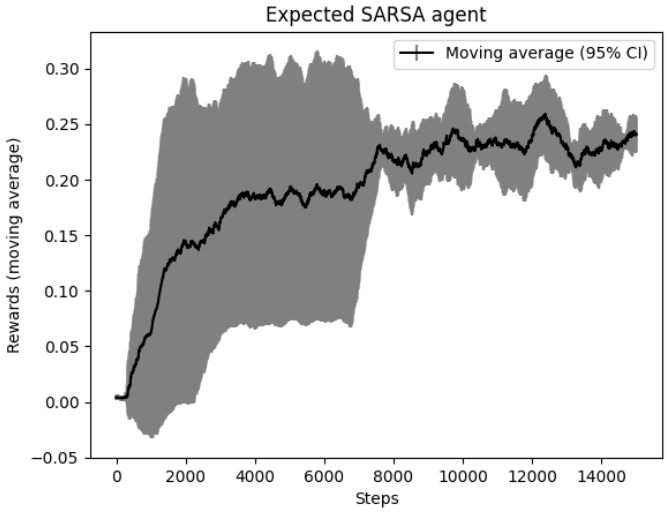

Figure 9: Average moving reward for expected SARSA in RiverSwim environment

B Q-values and policies for all agents

B.1 Q-learning agent

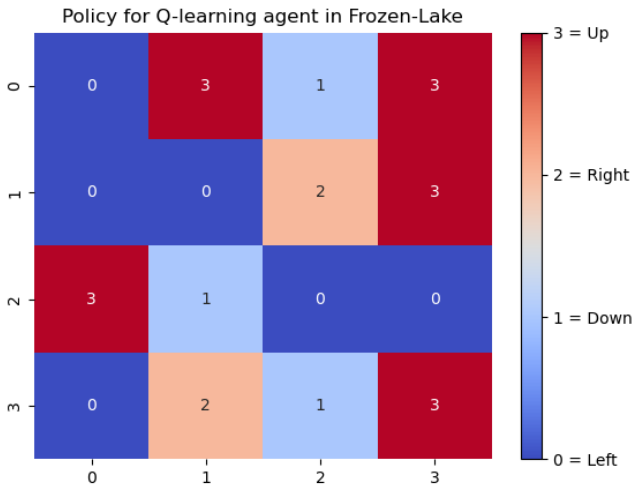

Figure 10: Policy for Q-learning agent in the Frozen lake environment.

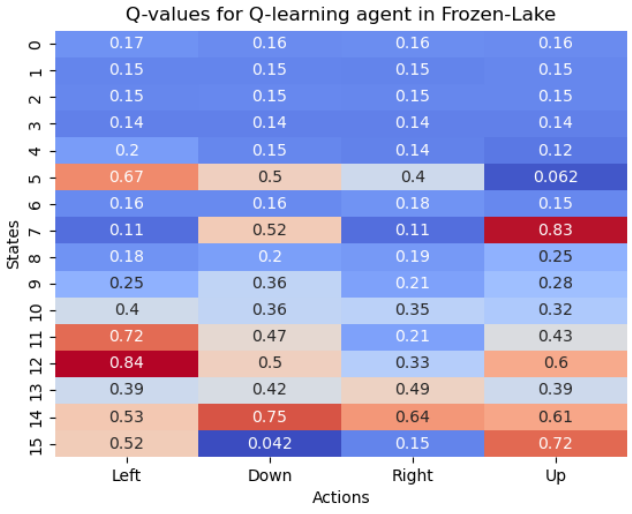

Figure 11: Q-values for Q-learning agent in the Frozen lake environment.

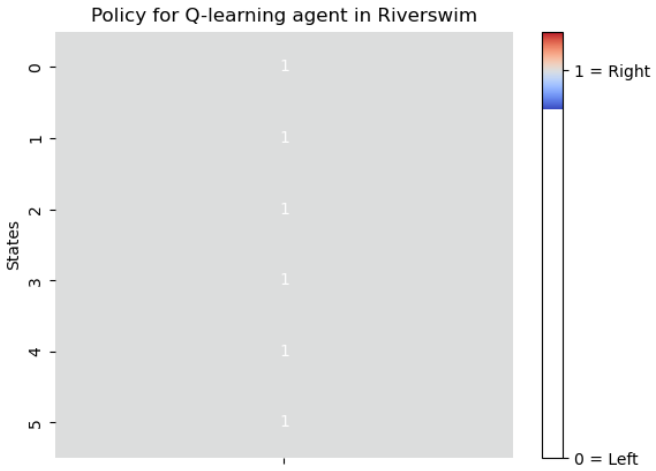

Figure 12: Policy for Q-learning agent in the Riverswim environment.

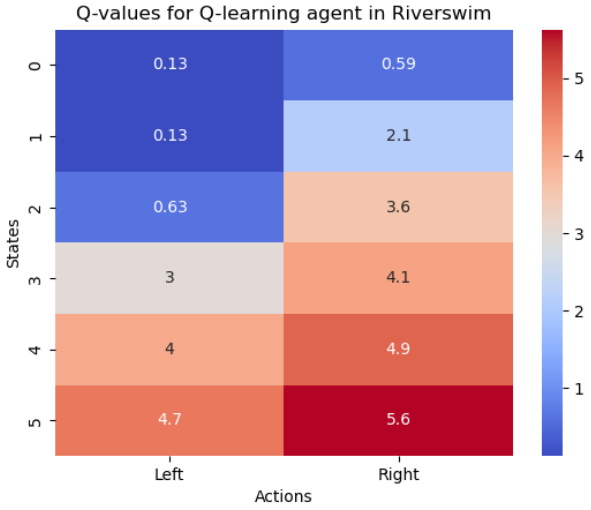

Figure 13: Q-values for Q-learning agent in the Riverswim environment.

B.2 Double Q-learning agent

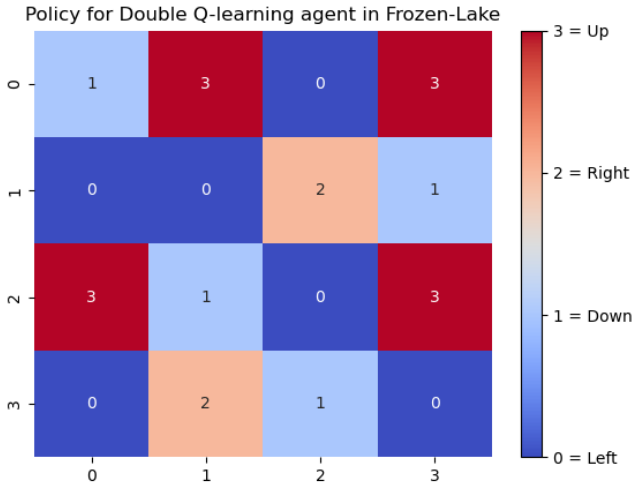

Figure 14: Policy for double Q-learning agent in the Frozen lake environment.

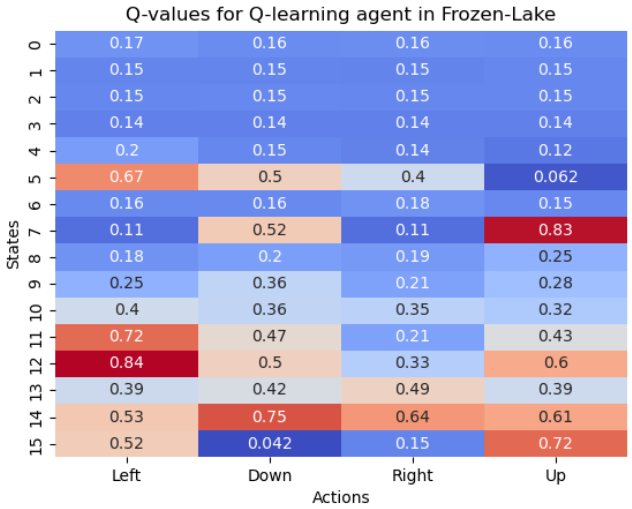

Figure 15: Q-values for double Q-learning agent in the Frozen lake environment.

Figure 16: Policy for double Q-learning agent in the Riverswim environment.

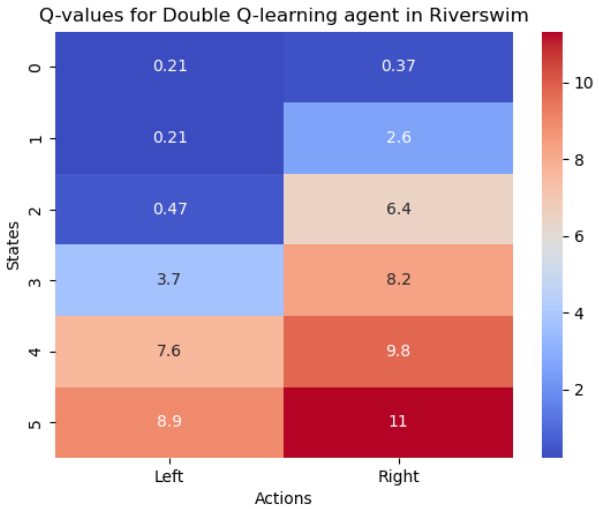

Figure 17: Q-values for double Q-learning agent in the Riverswim environment.

B.3 SARSA agent

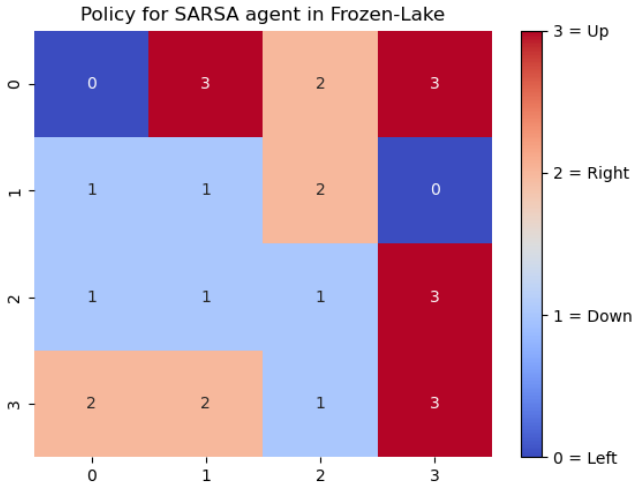

Figure 18: Policy for the SARSA agent in the Frozen lake environment.

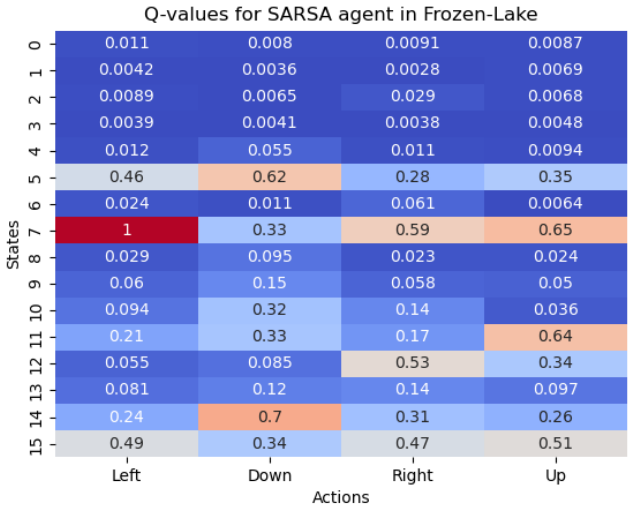

Figure 19: Q-values for the SARSA agent in the Frozen lake environment.

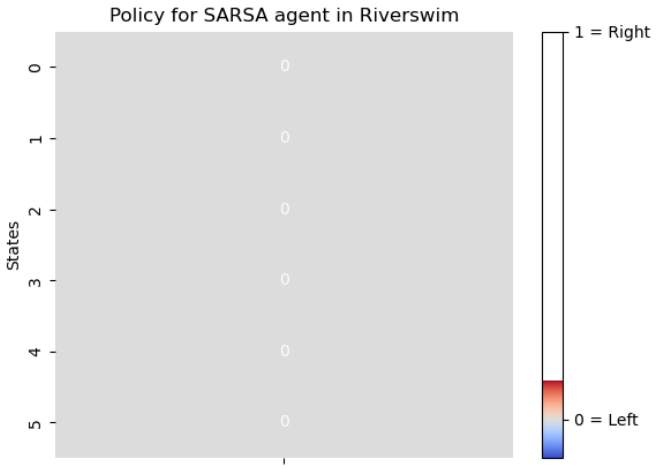

Figure 20: Policy for the SARSA agent in the Riverswim environment.

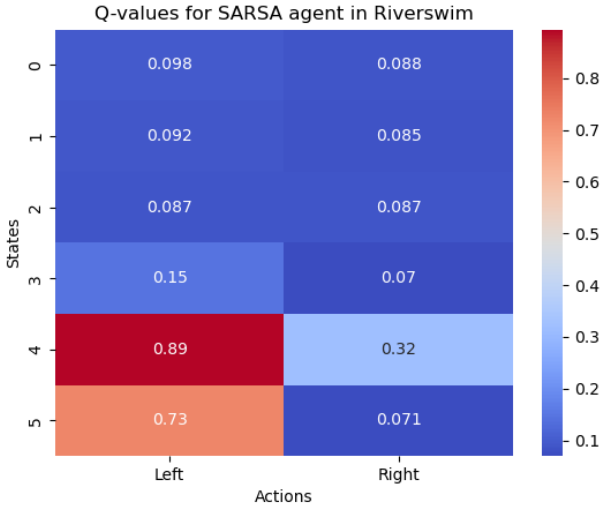

Figure 21: Q-values for the SARSA agent in the Riverswim environment.

B.4 Expected SARSA agent

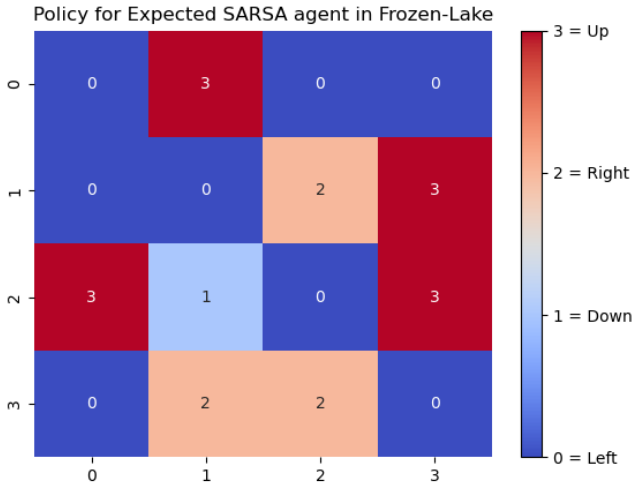

Figure 22: Policy for the expected SARSA agent in the Frozen lake environment.

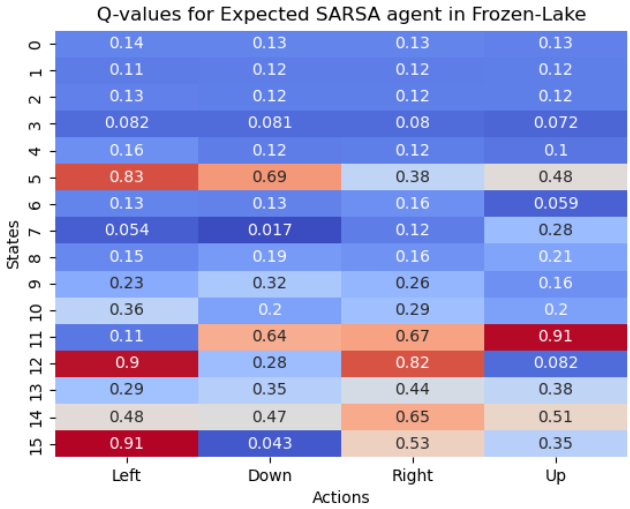

Figure 23: Q-values for the expected SARSA agent in the Frozen lake environment.

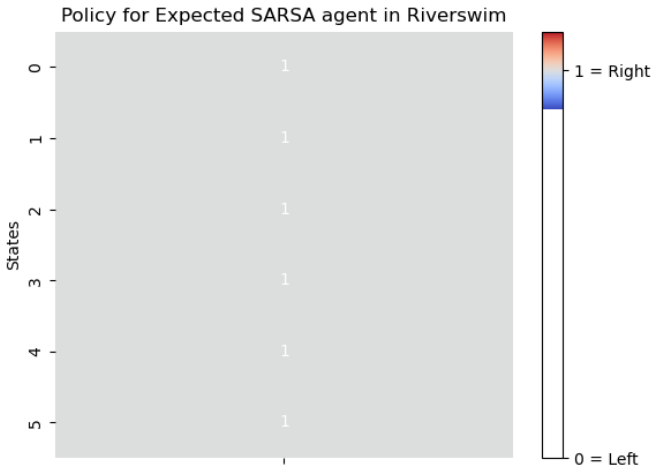

Figure 24: Policy for the expected SARSA agent in the Riverswim environment.

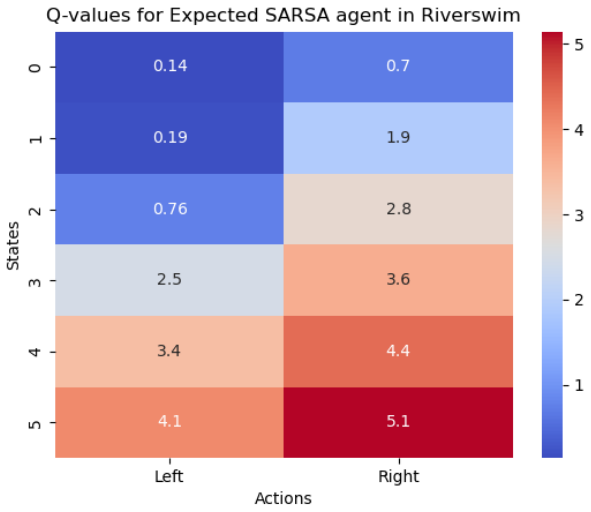

Figure 25: Q-values for the expected SARSA agent in the Riverswim environment.