Assignment 2 - Report

Kamil Topolewski

1 Reinforcement Learning

Reinforcement learning(RL) is an area of Machine Learning(ML). It is similar to the supervised learning technique since the concept of feedback is present as well. However, reinforcement learning can solve problems where labelling all data points might be infeasible. This ML technique introduces a notion of rewards. The algorithm learns the expected rewards by exploring different actions and then exploiting that information to find the most optimal path. Therefore, instead of looking for patterns in data (like supervised and unsupervised learning), RL is trying to maximize the reward signal.

Reinforcement model makes a sequence of decisions. Choices are made based on Q values which are expected rewards for each action taken with respect to a state. True expected rewards are usually not known, therefore the values are estimated based on past observations. The expected Q values are then updated after each new observation using formula 1.

$$\Delta Q(s,a) = \eta(r - Q(s,a)) \tag{1}$$

where *Q(s,a)* is the expected reward for state s and action a, η is the learning rate and r is the reward.

Methods for choosing actions with respect to Q values are called policies. Dependent on the intent, a different policy is chosen. Popular methods are:

- **Greedy.** Always chooses the action with the maximum Q value, represented with formula 2. However, selecting actions with the highest Q values every time might cause insufficient exploring. The agent is likely to repeat the same routes without investigating other (potentially more optimal) paths.

$$Q(s,a^*) > Q(s,a_j) \tag{2}$$

  where *Q(s,a)* is the expected reward for state s and chosen action a^*, j is the index of all other actions

- **ϵ-Greedy.** Uses the standard greedy policy with probability $1 - \epsilon$ and random action the rest of the time. Randomness enforces exploration of the environment. The trade-off balance between exploration and exploitation is parameterized with ϵ.

- **Softmax.** For each state, this policy encodes the actions as probabilities. Actions with higher expected reward are given higher probability, therefore are chosen more often. The probabilities are calculated with following function:

$$P(a) = \frac{e^{Q(s,a)/\tau}}{\sum_b e^{Q(s,b)/\tau}} \tag{3}$$

  where *Q(s,a)* is the expected reward for state s and action a, τ is parameter controlling the distribution (high τ flattens it) and b is all possible actions.

- **Optimistic greedy.** Initializes the expected rewards to a high value and uses the greedy policy. The actions that have not been tested will have high value, therefore, the algorithm will favour them. Then through the process of exploration, the expected rewards will adjusted based on the true rewards.

2 Future Rewards

The immediate rewards in each state might not always be available. For instance, in games like

mazes, the positive feedback is usually a few steps into the future. Therefore, the explained RL model will not learn any expected values apart from states prior to finish. SARSA and Q-learning are algorithms which solve this problem through future rewards. This takes into account the expected value of the current state, as well as the next chosen action. Therefore, the algorithm learns the Q for the future state plus the current reward. This change in Q values is represented with formula 4.

$$\Delta Q(s,a) = \eta[r - (Q(s,a) - \gamma Q(s',a'))] \quad (4)$$

where *Q(s',a')* is the expected reward for next state s' and future action a', η is the learning rate, r is the reward and γ is a discount factor.

The γ is a factor which discount the future reward. This encourages the algorithm to take a shorter path when the reward is similar. The importance of the future reward in comparison to the immediate is depends on the parameter γ. For "homing" based tasks, where for each step the reward is negative, the γ is not necessary, since the future reward is already discounted by taking an extra step.

3 SARSA and Q-learning

SARSA and Q-learning algorithms iteratively implement the future rewards method to find the best path. SARSA is called an on-policy technique which means it uses the same policy for choosing all actions. Therefore, Q-values for state s and action a are updated using $Q(s', a')$ selected under the same policy. The update rule has been shown in Formula 4, where both expected rewards are chosen using the same policy.

Whereas the Q-learning is an example of an off-policy algorithm. The expected rewards are updated using the Q-values of the next state and action chosen by a greedy algorithm. The altered update equation is shown in Formula 5

$$\Delta Q(s,a) = \eta[r - (Q(s,a) - \gamma\, max_a Q(s',a'))] \quad (5)$$

where *Q(s',a')* is the expected reward for next state s' and future action a', η is the learning rate, r is the reward, γ is a discount factor and max_a is an action maximizing the expected reward.

4 Evaluation

Mentioned RL algorithms will be evaluated on the cliff walking "homing" task[1]. Every movement to an empty tile gives a reward of $r = -1$. Stepping on a cliff tile gives a reward $r = -100$ and ends the trial. The game also finishes when the agent reaches the final square.

Firstly SARSA and Q-learning algorithms will be tested. Then different settings for ϵ-greedy policy and learning rates will be evaluated.

4.1 Algorithms

For evaluation of cliff walking task, SARSA and Q-learning algorithm have been developed. On-policy SARSA uses ϵ-greedy policy for both current and future actions. For better comparison, Q-learning uses the same ϵ-greedy policy for the current actions (and as defined greedy for the future).

In Appendix A, the learning curve for both algorithms has been plotted. A learning curve is a graph showing how total reward changes throughout the algorithm. At the very beginning of its exploration, the algorithm steps on cliff tiles early in the run, which results in the total reward being around -100. Then the accumulative reward decreases because it manages to explore some environment but ultimately steps on another unexplored cliff. Then avoiding the cliffs efficiently searches for the most optimal route. In around 150 trials both SARSA and Q-learning have "stopped learning". Due to the randomness caused by *epsilon*-greedy algorithm the performance is not stable. The standard errors for the algorithms have been plotted with shaded error bars.

On average, Q-learning results in lower reward compared to the SARSA algorithm. To understand the reason behind it, the most frequently chosen paths have been examined. This is shown using heat maps in Appendix B. The heat maps represent how many times agent

entered each state on average. The Q-learning tends to choose the most optimal path, which is alongside the cliff. This is due to the off-policy method which chooses the greedy future actions. This does not take into the account possibility of "falling" off the cliff. Following this path, the total reward will be potentially maximum, however, due to the nature of ϵ-greedy policy, there is a risk of stepping into the cliff tile. For $\epsilon = 0.1$, on average, the random action will be taken 10% of the time. Stepping alongside the edge gives 25% chance of stepping into the cliff, which is equivalent to 2.5% chance of "falling" with each movement. Therefore, despite finding the most optimal route, it results in a high standard error and decreased total reward performance visible on the graph in Appendix A.

According to the heat map in Appendix B, SARSA tends to choose the safe route. This is due to ϵ-greedy policy for future expected rewards. With ϵ probability, it will factor in stepping on the cliff tile in the future move. which will discourage the algorithm to choose those actions in the next trails. That results in taking a "safer" route, further away from the cliff.

4.2 Learning rate

The learning rate is a parameter controlling the significance of the weight update. The average learning curve of Q-learning and SARSA (under ϵ-greedy policy with ϵ=0.1) for different learning rates has been plotted in Appendix F. For increasing learning rates the convergence of the functions happens faster. For the learning rates [0.1, 0.25, 0.5] the average reward after convergence is very similar. However, the lower learning rates take more trials. The significance of weight change is smaller which results in slower learning. For a large learning rate of 0.9, the algorithm learns quickly however the performance decreases over time. The magnitude of updates is so substantial that causes route changes after convergence. This is visualized with heat maps in Appendix G. The heat map shows a relatively uniform distribution in large portions of the grid, which means that the agent does not follow the same path frequently. Whereas for the other learning rates a clear, most often used path can be distinguished.

The Q-learning shows similar relation of faster training with a higher learning rate. However, even for a large learning rate, after the convergence the performances are similar. Since the future rewards are chosen with a greedy method, there are less random actions chosen. This allows Q-learning to be more resilient to the high weight changes.

4.3 ϵ-greedy

As mentioned in Section 1, ϵ-greedy is a policy where the ϵ controls the trade-off between exploration at exploitation. Higher ϵ increases the probability of choosing a random direction which enforces vaster exploration. To find the optimal value of ϵ the SARSA algorithm has been tested with different settings of ϵ-greedy policy. Results are presented in Appendix C. Higher exploration yields a lower average reward. That is due to a higher possibility of making a random step which in the long run decreases the performance. With decreasing epsilon the average reward increases and becomes more "stable". When no randomness is introduced, the algorithm yields the best result. This is due to the fact that weights are initialized to zero and all rewards are negative. That essentially turns ϵ-greedy policy into an optimistic-greedy. The explored states will be adjusted with a negative reward, which will enforce picking the action towards unvisited tiles. Eventually after exploring the entire environment, by always picking greedy actions, the algorithm follows the most optimal path exactly every time. The algorithms using higher epsilon tend to choose safer paths and be more chaotic due to the randomness. The paths chosen are plotted on a heat map in Appendix D.

Similar performance relation can be observed for Q-learning. With increasing epsilon, the algorithm's average reward is smaller. However, with Q-learning, the decrease in performance is more significant. That occurs because the Q-learning will always choose the most optimal route. Therefore, walking along the cliff, with a higher chance of making a random action,

increases the chances of "falling off" a cliff. This is visualized in Appendix E. The routes are similar, however, agents with higher ϵ misstep from the path more often. For zero ϵ, the method again becomes a variation of optimistic-greedy policy. Therefore, SARSA and Q-learning algorithm for ϵ=0 become the same algorithms and performs best.

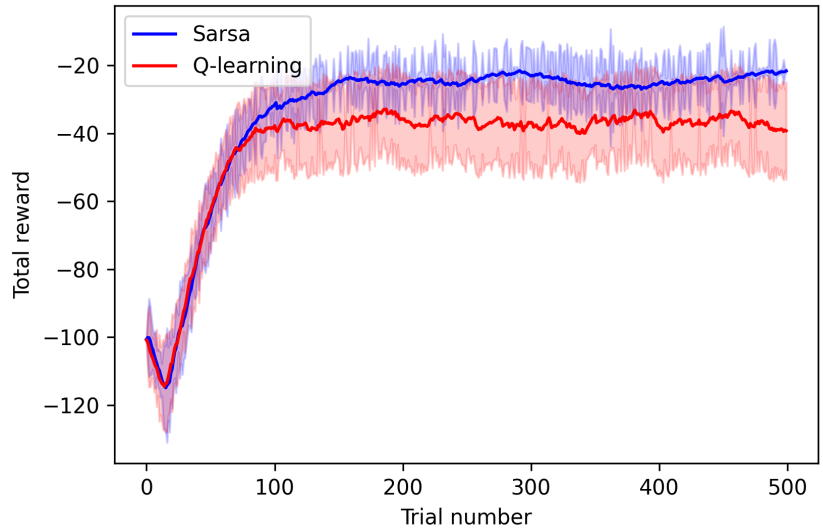
So is the ϵ-greedy always worse? The non-zero ϵ-greedy policy performs better in an environment where the distribution of rewards is noisier, because finding the optimal path takes more exploration[1]. Another advantage of the ϵ-greedy method is adapting to the change of rewards. After being trained, the greedy policy only checks expected values adjacent to its previously found optimal route.

References

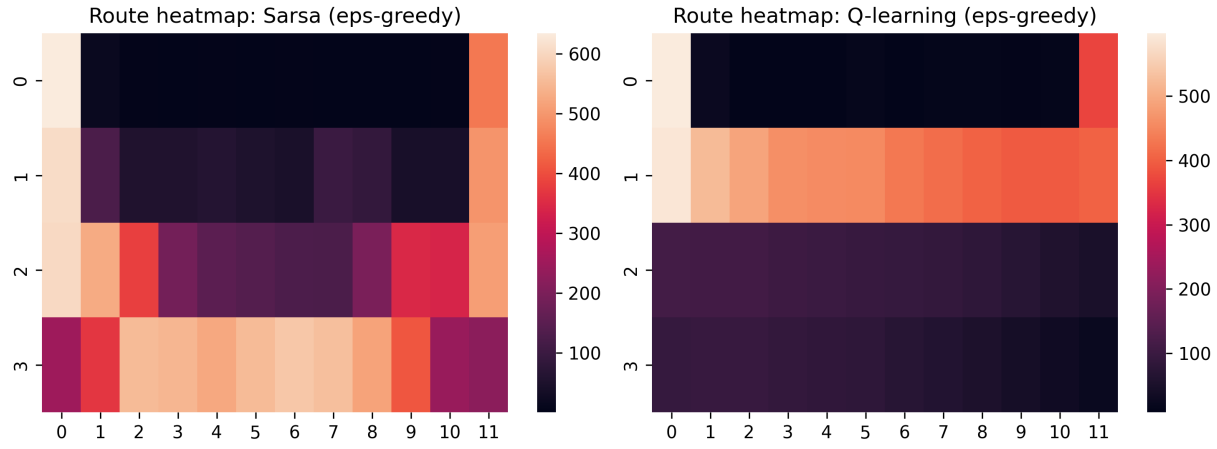
[1] R. Sutton and A. Barto, "Reinforcement learning: An introduction," 2018.

Appendix

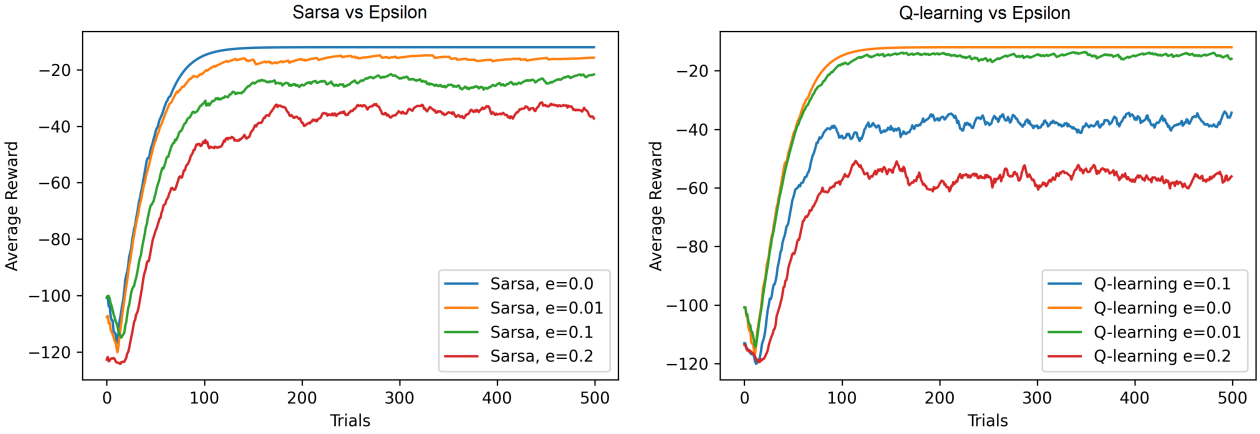
A. Average learning curve of Q-learnig and SARSA with shaded error-bars.



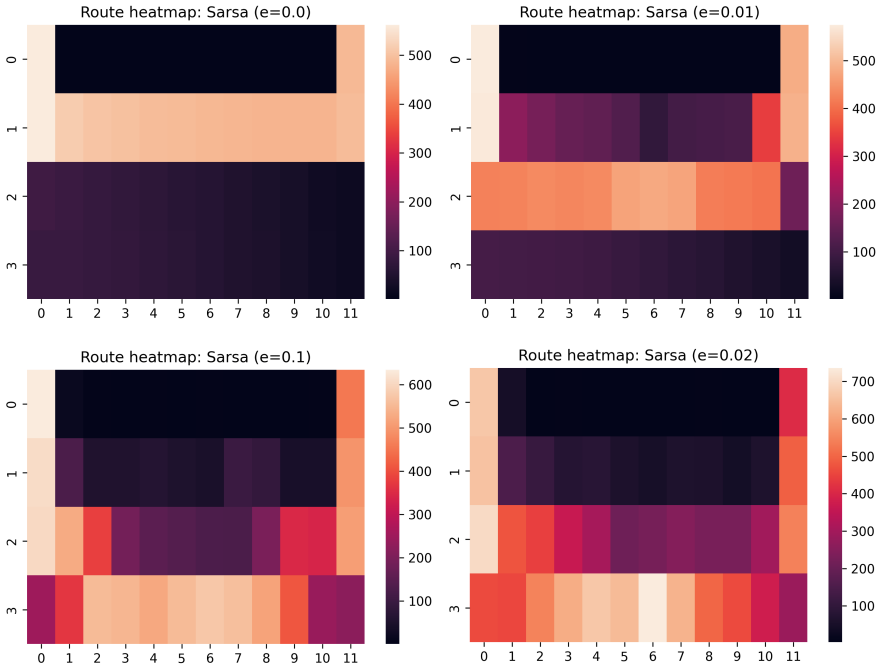
B. Heat maps of amount of times each state has been visited within 500 trails for SARSA and Q-learning.



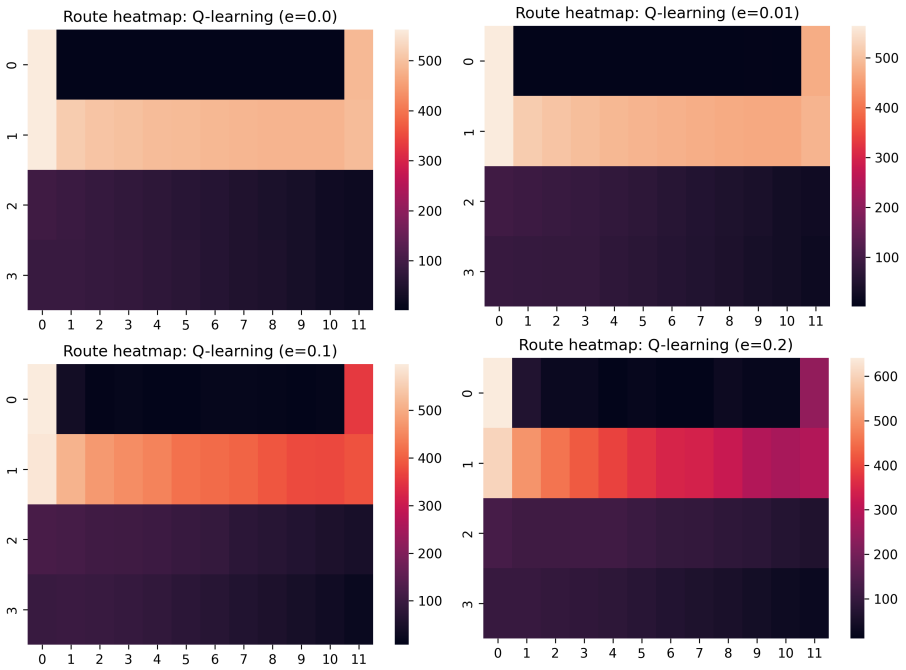
C. Average learning curve of Q-learnig and SARSA for different epsilon.



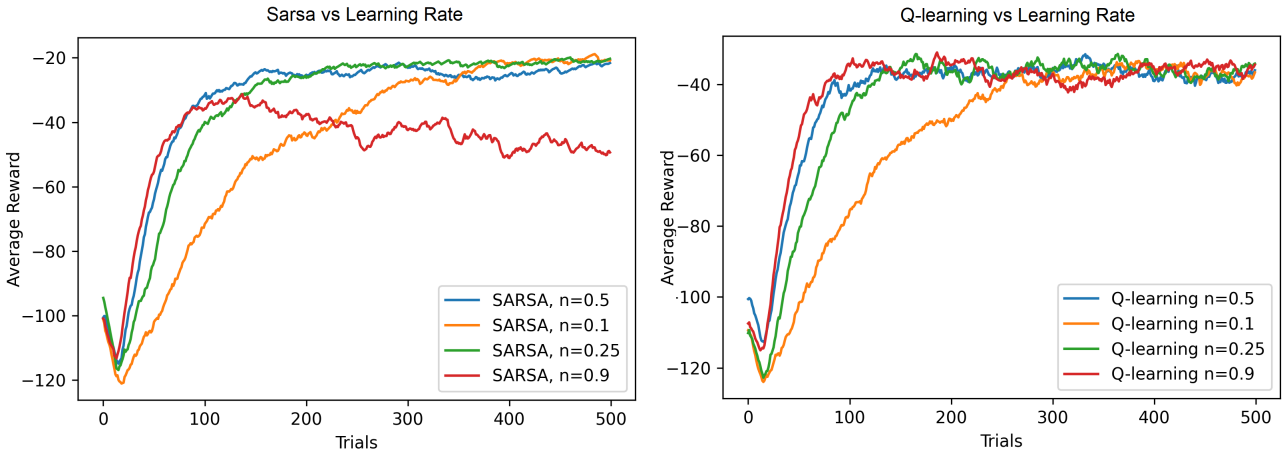
D. Heat maps of amount of times each state has been visited within 500 trails for SARSA under ϵ-greedy policy with different ϵ values



E. Heat maps of amount of times each state has been visited within 500 trails for Q-learning under ϵ-greedy policy with different ϵ values



F. Average learning curve of Q-learnig and SARSA for different learning rates under ϵ-greedy policy with ϵ=0.1



G. Heat maps of amount of times each state has been visited within 500 trails for SARSA under ϵ-greedy policy with different learning rates values.