COMP532-2021-22

Assignment 1 – Reinforcement Learning

Group Members	Email	Student ID
Huy Pham	Huycam1997@gmail.com	201534475
Pamela Mercado	p.mercado@liverpool.ac.uk	201309392
Utsav Dihingia	u.dihingia@liverpool.ac.uk	201599488
Saeth Wannasuphoprasit	s.wannasuphoprasit@liverpool.ac.uk	201585689

Problem 1

In Reinforcement Learning, an agent learns by receiving a reward after taking an action. In order to learn how to find the best actions, the agent normally runs through the same environment several times. In exploration, the agent attempts different actions to discover the received rewards. It must explore to make better action selections in the future. To maximize the number of rewards, the agent has to exploit what it has already experienced. If an agent spends on exploration too much, it can not stick to a path and exploit its knowledge. As a result, it is not really learning. However, without exploration, the optimal actions will not be found. Therefore, a balance between exploration and exploitation is important in Reinforcement Learning.

We can see from the plots below that epsilon = 0.1 is the perfect balance between exploitation and exploration since it has the highest optimal action and maximizes the average rewards whereas in the case of greedy (epsilon = 0) and epsilon = 0.01 there is too much exploitation respectively which drastically decreases the optimal action since they are relying too much on the safest action from the information the agent already has. Moreover, the value of epsilon is not proportionate with the number of received rewards. If we spend too much time on exploring, we might receive less number of rewards.

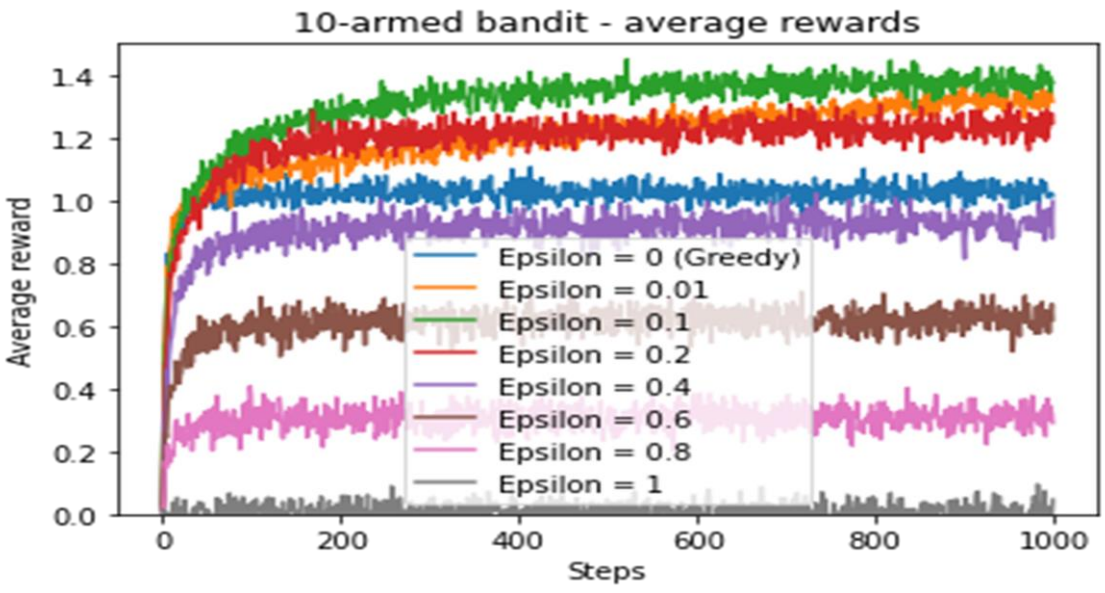


Figure 1. The number of average rewards using different epsilon values.

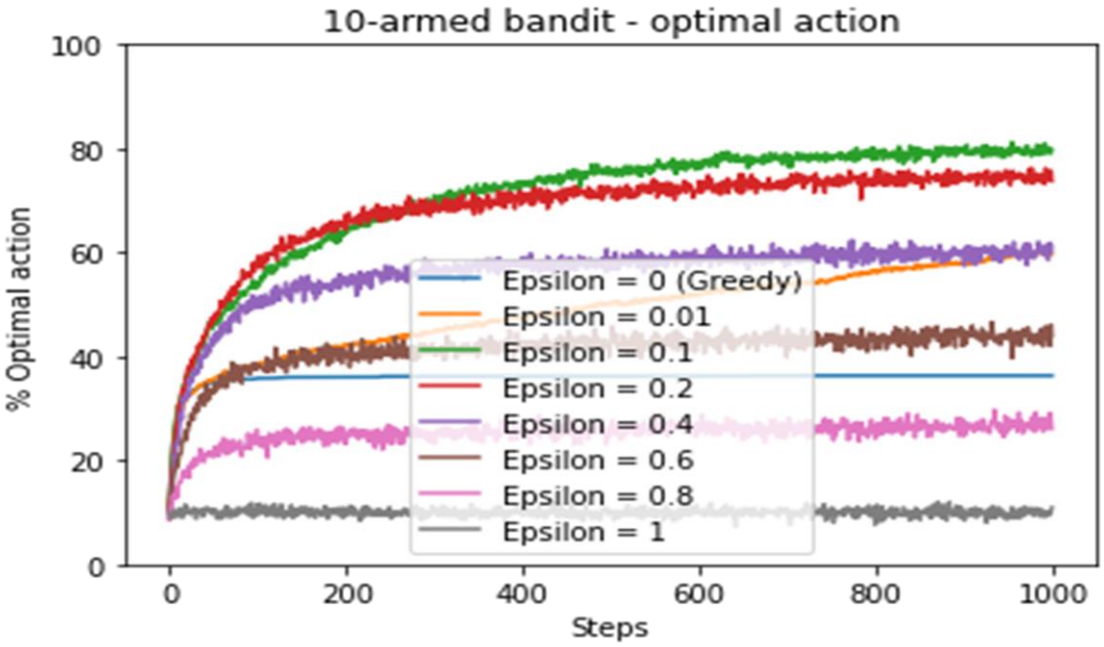
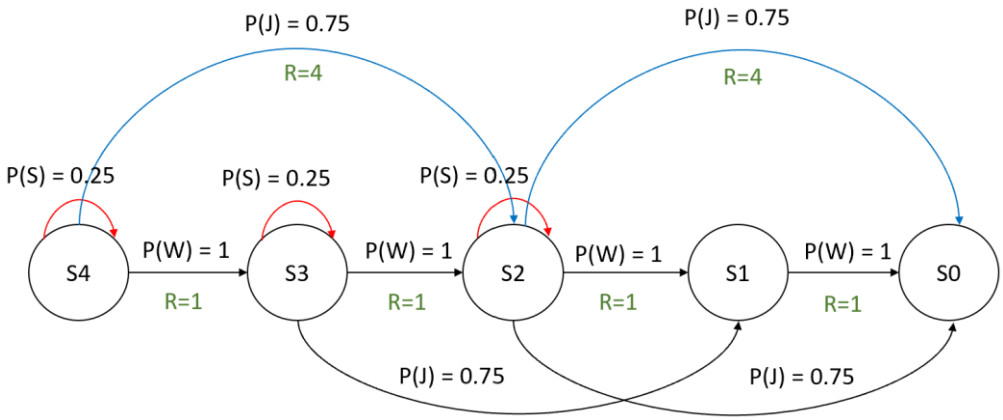


Figure 2. Percentage of optimal action versus steps with different epsilon values

Problem 2

Markov Decision Process Diagram



Compute $V^*(2)$:

$$\mathbf{V^*(s)} = \sum \boldsymbol{P(s'|s,a)[R(s',a,s) + \gamma V^*(s')}$$

$$\boldsymbol{\gamma = 0.5}$$

V*(S0) = 0

V*(S1) = $\boldsymbol{P(S_0|S_1, walk)[R(S_0, walk, S_1) + \gamma V^*(S_0)}$

$= 1 \times (1 + 0.5 \times 0) = 1$

V*(S2) = $\max_{a \in A(s)} \sum P(s'|s,a)[R(s',a,s) + \gamma V^*(s')$

$$= \max \begin{cases} S_2 \to S_1 \\ S_2 \to S_2 \; or \; S_2 \to S_0 \end{cases}$$

$$= \max \begin{cases} P(s_2|s_1, walk)[R(s_2, walk, s_1) + \gamma V^*(s_1) \\ P(s_2|s_2, stay)[R(s_2, stay, s_2) + \gamma V^*(s_2) + \; P(s_0|s_2, jump)[R(s_0, jump, s_2) + \gamma V^*(s_0) \end{cases}$$

$$= \max \begin{cases} 1 \times (1 + 0.5 \times 1) = 1.5 \\ 0.25 \times \big(0 + 0.5V^*(S_2)\big) + 0.75 \times (4 + 0.5 \times 0) \end{cases}$$

$$= \max \begin{cases} 1.5 \\ 0.125V^*(S_2) + 3 \end{cases}$$

If $0.125V^*(S_2) + 3$ is max

Then, $V^*(S_2) = \; 0.125V^*(S_2) + 3$

$$0.875V^*(S_2) = 3$$

=> $\boldsymbol{V^*(S_2) \approx 3.429}$

Compute Q*(3,Jump):

V*(S3) = $\max_{a \in A(s)} \sum P(s'|s,a)[R(s',a,s) + \gamma V^*(s')$

$$= \max \begin{cases} S_3 \to S_2 \\ S_3 \to S_3 \; or \; S_3 \to S_1 \end{cases}$$

$$= \max \begin{cases} P(s_3|s_2, walk)[R(s_3, walk, s_2) + \gamma V^*(s_2) \\ P(s_3|s_3, stay)[R(s_3, stay, s_3) + \gamma V^*(s_3)] + \; P(s_1|s_3, jump)[R(s_1, jump, s_3) + \gamma V^*(s_1)] \end{cases}$$

$$= \max \begin{cases} 1 \times (1 + 0.5 \times 3.429) = 2.7145 \\ 0.25 \times \big(0 + 0.5V^*(S_3)\big) + 0.75 \times (4 + 0.5 \times 1) \end{cases}$$

$$= \max \begin{cases} 2.7145 \\ 0.125V^*(S_3) + 3.375 \end{cases}$$

If $0.125V^*(S_2) + 3.375$ is max

Then, $V^*(S_3) = \; 0.125V^*(S_3) + 3.375$

$$0.875V^*(S_2) = 3.375$$

=> $\boldsymbol{V^*(S_3) \approx 3.857}$

Problem 3

Episode 1: {1, 3, 5, 4, 2, 7}

Episode 2: {2, 3, 5, 6, 4, 7}

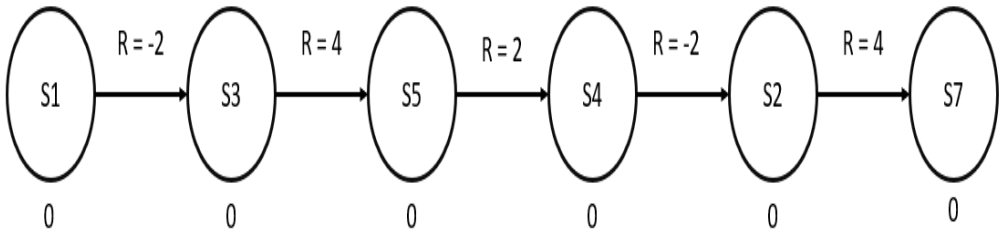
Episode 3: {5, 4, 2, 7}

$$\boldsymbol{V_{(st)} \leftarrow V_{(st)} + \alpha[R_{t+1} + \gamma V_{(s_{t+1})} - V_{(st)}]}$$

$$\boldsymbol{\gamma = 1}$$

$$\boldsymbol{\alpha = 0.5}$$

Episode 1



State	1	3	5	4	2	7
$V_{(st)}$	0	0	0	0	0	0
$V_{(st)} \leftarrow V_{(st)}$	-1	2	1	-1	2	0

$$\boldsymbol{V_{(s1)} \leftarrow V_{(s1)} + \alpha[R_{s1 \rightarrow s3} + \gamma V_{s3} - V_{s1}]}$$

$$\leftarrow 0 + 0.5 \times (-2 + 1 \times 0 - 0)$$

$$\leftarrow -1$$

$$\boldsymbol{V_{(s3)} \leftarrow V_{(s3)} + \alpha[R_{s3 \rightarrow s5} + \gamma V_{s5} - V_{s3}]}$$

$$\leftarrow 0 + 0.5 \times (4 + 1 \times 0 - 0)$$

$$\leftarrow 2$$

$$\boldsymbol{V_{(s5)} \leftarrow V_{(s5)} + \alpha[R_{s5\rightarrow s4} + \gamma V_{s4} - V_{s5}]}$$

$$\leftarrow 0 + 0.5 \times (2 + 1 \times 0 - 0)$$

$$\leftarrow 1$$

$$\boldsymbol{V_{(s4)} \leftarrow V_{(s4)} + \alpha[R_{s4\rightarrow s2} + \gamma V_{s2} - V_{s4}]}$$

$$\leftarrow 0 + 0.5 \times (-2 + 1 \times 0 - 0)$$

$$\leftarrow -1$$

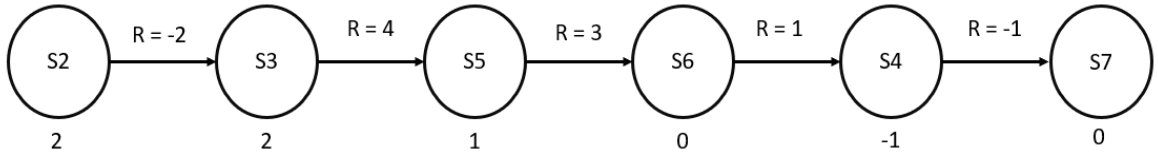
$$\boldsymbol{V_{(s2)} \leftarrow V_{(s2)} + \alpha[R_{s2\rightarrow s7} + \gamma V_{s7} - V_{s2}]}$$

$$\leftarrow 0 + 0.5 \times (4 + 1 \times 0 - 0)$$

$$\leftarrow 2$$

$$\boldsymbol{V_{(s7)} \leftarrow 0}$$

Episode 2



State	2	3	5	6	4	7
$V_{(st)}$	2	2	1	0	-1	0
$V_{(st)} \leftarrow V_{(st)}$	1	3.5	2	0	-1	0

$$\boldsymbol{V_{(s2)} \leftarrow V_{(s2)} + \alpha[R_{s2\rightarrow s3} + \gamma V_{s3} - V_{s2}]}$$

$$\leftarrow 2 + 0.5 \times (-2 + 1 \times 2 - 2)$$

$$\leftarrow 1$$

$$\boldsymbol{V_{(s3)} \leftarrow V_{(s3)} + \alpha[R_{s3\rightarrow s5} + \gamma V_{s5} - V_{s3}]}$$

$$\leftarrow 2 + 0.5 \times (4 + 1 \times 1 - 2)$$

$$\leftarrow 3.5$$

$$\boldsymbol{V_{(s5)} \leftarrow V_{(s5)} + \alpha[R_{s5\rightarrow s6} + \gamma V_{s6} - V_{s5}]}$$

$$\leftarrow 1 + 0.5 \times (3 + 1 \times 0 - 1)$$

$$\leftarrow 2$$

$$\boldsymbol{V_{(s6)} \leftarrow V_{(s6)} + \alpha[R_{s6\rightarrow s4} + \gamma V_{s4} - V_{s6}]}$$

$$\leftarrow 0 + 0.5 \times (1 + 1 \times -1 - 0)$$

$$\leftarrow -0$$

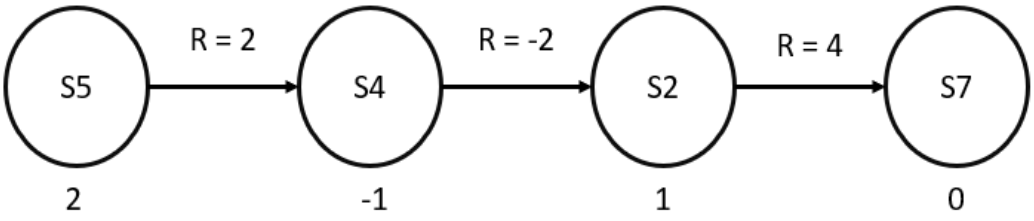
$$\boldsymbol{V_{(s4)} \leftarrow V_{(s4)} + \alpha[R_{s4\rightarrow s7} + \gamma V_{s7} - V_{s4}]}$$

$$\leftarrow -1 + 0.5 \times (-1 + 1 \times 0 - -1)$$

$$\leftarrow -1$$

$$\boldsymbol{V_{(s7)} \leftarrow 0}$$

Episode 3



State	5	4	2	7
$V_{(st)}$	2	-1	1	0
$V_{(st)} \leftarrow V_{(st)}$	1.5	-1	2.5	0

$$\boldsymbol{V_{(s5)} \leftarrow V_{(s5)} + \alpha[R_{s5\rightarrow s4} + \gamma V_{s4} - V_{s5}]}$$

$$\leftarrow 2 + 0.5 \times (2 + 1 \times -1 - 2)$$

$$\leftarrow 1.5$$

$$\boldsymbol{V_{(s4)} \leftarrow V_{(s4)} + \alpha[R_{s4\rightarrow s2} + \gamma V_{s2} - V_{s4}]}$$

$$\leftarrow -1 + 0.5 \times (-2 + 1 \times 1 - -1)$$

$$\leftarrow -1$$

$$\boldsymbol{V_{(s2)} \leftarrow V_{(s2)} + \alpha[R_{s2 \rightarrow s7} + \gamma V_{s7} - V_{s2}]}$$

$$\leftarrow 1 + 0.5 \times (4 + 1 \times 0 - 1)$$

$$\leftarrow 2.5$$

$$\boldsymbol{V_{(s7)} \leftarrow 0}$$

Problem 4

a) **What does the Q-learning update rule look like in the case of a stateless or 1-state problem? Clarify your answer.**

N-armed bandits are single state or stateless. There is no exploration as there is no state/only 1 state so only one possible path. Values in Q-table can be initialized to 0. For stateless problem, the values after the Q-learning update rule would remain 0 as there is no interaction with the environment. Stateless problems are armed bandits where the actions do not affect the state.

For stateless environment the Q-learning update rule only depends on action-reward pair. So, depending on different pairs of actions it will award different rewards.

1-state n-armed bandit problems means that no matter what action you take, the agent stays in the same state. The reward depends on the action you take.

For a 1-state environment the Q-learning update rule will have no next state or new state as there is only one state. The agent will stay in the same state but will be awarded different rewards for different actions.

b) **Discuss the main challenges that arise when moving from single- to multi-agent learning, in terms of the learning target and convergence.**

Non-stationary transitions in multi-agent learning are a problem as there are multiple agents so the state becomes a joint state for all agents. Therefore, this results in transitions becoming dependent on the actions of all agents instead of one, as to succeed, agents must consider the behaviour of other agents and adapt to the joint behaviour. The Markov property is no longer suitable as it only refers to the reward and present state of an individual agent. Therefore, the algorithms would fail to converge to optimal solutions as the environment is no longer stationary from each agent's perspective as each agent would end up entering an endless cycle of adapting to other agents. In single-agent reinforcement learning, stationary transition probability values stay the same, however, training multi-agents is more of a challenge as transition probabilities change over time as they do not have as much information about the environment compared to single-agents. This results in variation in the expected return for a state-action pair of a given agent where values may be inaccurate. Partial observability is another challenge as the agents having less information about the states can lead to learning suboptimal policies. For example, in cooperative environments, if

other agents learn a useful policy, an agent can become “lazy” and cannot learn as it stops exploring to not affect the return.

All agents’ actions influence the return; therefore, it is difficult for an individual agent to separate its own success to the team’s success. If an agent chooses an action that maximises reward in each state, the return could still be negative if all the other agents took exploratory actions. Therefore, the agent could incorrectly adjust its policy to reduce the probability of selecting that action. This is a credit assignment problem which would cause challenges in the learning progress.

Problem 5

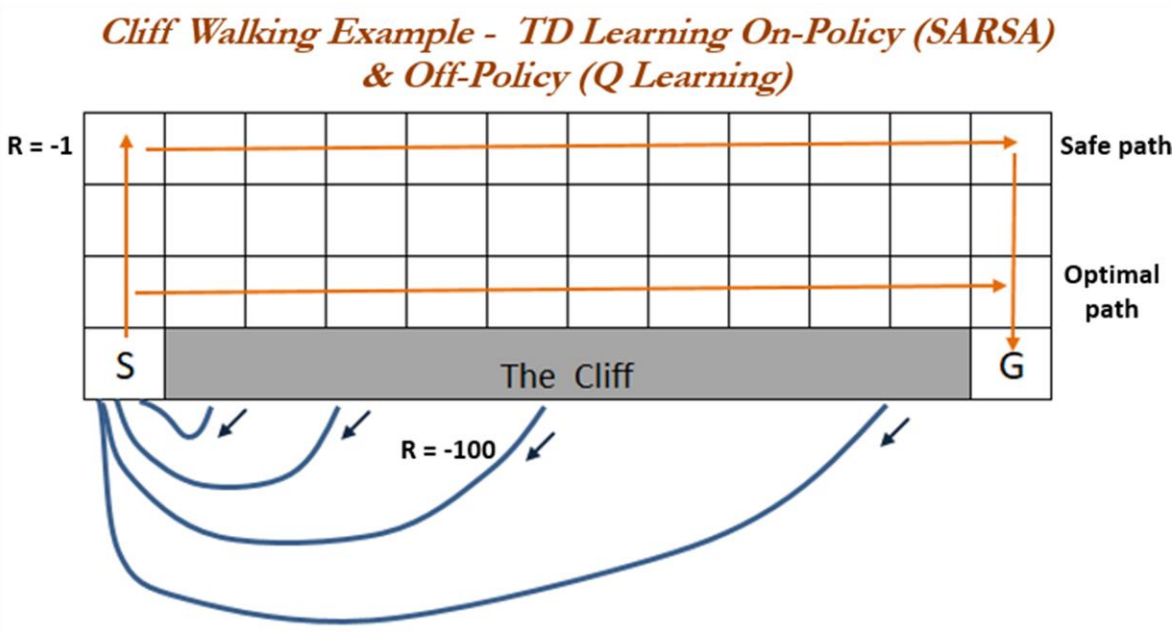


Figure 4. The pathway the agent takes in the Q-learning and Sarsa method.

The agent followed safe path when using Sarsa method and followed optimal path when using Q-learning.

We choose epsilon = 0.1 for both as referenced from the book Sutton and Barto. From the plot we can see the Sarsa converges earlier than the Q-learning. We can say that the agent learns faster using Q-learning than Sarsa but has a higher risk of falling off the cliff. The Sarsa algorithm should be mainly used in situations where the agent's performance and experience are cared for.

During its learning process, the agent chooses the safest path rather than choosing a riskier path near the cliff whereas the Q-learning algorithm is used where the agent's performance is not cared for and just wants to learn at an optimal greedy policy here the agent does not mind falling off the cliff sometimes due to randomness.

References

Canese L, Cardarilli GC, Di Nunzio L, Fazzolari R, Giardino D, Re M, Spanò S. Multi-Agent Reinforcement Learning: A Review of Challenges and Applications. *Applied Sciences*. 2021; 11(11):4948. https://doi.org/10.3390/app11114948 (Accessed: 3 March 2022).

Kouridi, C. (2020) 'A brief summary of challenges in Multi-agent RL', *Christina's blog*, 02/01. Available at: https://christinakouridi.blog/2020/01/02/challenges-multiagent-rl/ (Accessed: 3 March 2022).

Kumar, V. (2019) *Reinforcement learning: Temporal-Difference, SARSA, Q-Learning & Expected SARSA in python*. Available at: https://towardsdatascience.com/reinforcement-learning-temporal-difference-sarsa-q-learning-expected-sarsa-on-python-9fecfda7467e (Accessed: 7 March 2022).

Soemers, D. (2018) 'What is the difference mathematically?', What is the difference between Q-learning and SARSA? Available at: https://stackoverflow.com/questions/6848828/what-is-the-difference-between-q-learning-and-sarsa (Accessed: 7 March 2022).