# IN4320: Machine Learning Assignment 5

Zheng Liu (4798406)

May 17, 2019

# 1

I reproduced the result illustrated in assignment sheet, the result is shown in Figure 1. In total, there are 12 sweeps required to achieve the convergence, to be more specific, in the $11th$ step, the convergence result is already calculated, the $12th$ step is the confirmation of convergence that $Q_{i+1} = Q_i$ which is the optimal $Q^*$. The optimal policy $\pi^*$ is also shown in Figure 1, the red arrows demonstrate the optimal actions in certain states (in some states the optimal actions are not unique). The reason that value in state $G$ is 0 is that $G$ is already the terminal state, episode ends immediately when robot reaches here, there is not "reward $R^a_{ss'}$" or "value of next state $\gamma V^*(s')$", so there is no more move meaning the value here would be 0.

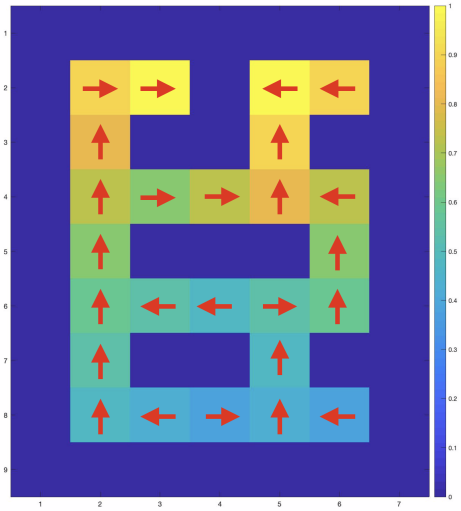

Figure 1: Optimal value function for $\gamma = 0.9$ and the optimal policy (red arrow)

# 2

The optimal value function $V^*$ for $\gamma = 0$, $\gamma = 0.1$, $\gamma = 0.5$ and $\gamma = 1$ is shown in Figure 2. It is obvious that with the increment of $\gamma$, the value in each state in the maze is equal or higher than the one with lower $\gamma$. In details, values of state $(2, 3)$ and $(2, 5)$, neighbour of $G$, for these four $\gamma$ are always 1. When $\gamma = 0$ only value of state $(2, 3)$ and $(2, 5)$ are not 0 and equal to 1 yet when $\gamma = 1$ all values in available states are 1. For different $\gamma$ they influence the optimal value functions $V^*$ thus influence the optimal policies $\pi^*$. Also compare with $\gamma = 0.9$ in Figure 1, we can see that for $\gamma = 0$, 0.1 and 0.5 it is always impossible for agent to achieve reward starting from state $S$ because the rewards starting in $S$ are 0, for $\gamma = 0.9$ and 1, they can achieve reward, but $\gamma = 0.9$ could urge agent obtain reward faster, but as there is not loss/punishment due to time for $\gamma = 1$, agent would wander in the maze. Generally for certain states, high $\gamma$ leads high reward and low $\gamma$ leads quick move, but when lower than some certain threshold value reward becomes 0.
The reason that $\gamma = 1$ will work here is that it could reach the final state and the

number of iteration is limited but not infinite meaning the process is convergent but not divergent.

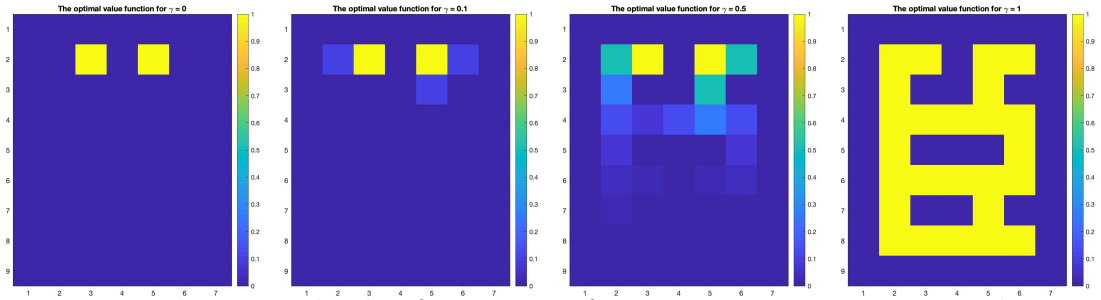

Figure 2: The optimal value function $V^*$ for $\gamma = 0$, $\gamma = 0.1$, $\gamma = 0.5$ and $\gamma = 1$

# 3

I implemented $Q$-learning with $\epsilon$-greedy exploration, setting a maximum number 2000 as steps limit per episode, when the robot reaches terminal state $G$ or the step limit it is reset to the starting state $S$.
For exploration $\epsilon = \{0.1, 0.3, 0.5, 0.7, 0.9\}$ and learning rate $\alpha = \{0.1, 0.3, 0.5, 0.7, 0.9\}$ were tried. I plotted the differences between the value function estimated by $Q$-learning and the true value function from $Q$-iteration in norm-2 with 10000 interactions with the system, the results are shown in Figure 3 and 4. There are 5 sub-figures in each figure, for Figure 3, $\alpha$ is fixed and 5 different lines apply 5 different $\epsilon$ values and for Figure 4, $\epsilon$ is fixed and 5 different lines apply 5 different $\alpha$ values. The $x$ axis is in logarithm scale.
Generally, for fixed $\alpha$, there is the trend that with the increment of $\epsilon$, the process converges faster with lower interaction numbers. The more interactions executed, the bigger differences are there between different $\epsilon$. For $\epsilon = \{0.7, 0.9\}$, the two higher values in the set, the result of them is close to each other. The explanation is that higher value of $\epsilon$ reveals higher randomness of choosing actions, this leads agent to explore environment more, the initially randomised "route" would be corrected with higher probability, then the convergence is faster, when $\epsilon$ is higher than a certain value, the randomness of choosing action is similar
Similarly for fixed $\epsilon$, higher value of $\alpha$ resulted in faster convergence (except $\epsilon = 0.1/0.3, \alpha = 0.9$). The higher the value is the faster convergence reached, the reason could be that learning rate $\alpha$ represent the magnitude of steps the agent executes toward the target, the higher the value is the bigger step agent takes thus faster convergence.
High value of $\alpha$ does not work well when $\epsilon$ is small. It is clear in Figure 3 and 4 when $\epsilon = 0.1/0.3, \alpha = 0.9$ they converge rather slow, even worse than the ones with low $\alpha$ value. The reason could be that low randomness is considered yet the agent exploit too fast towards the termination to explore the environment thus less new information achieved. To be intuitive, agent learns (exploits) fast with few exploration, ones it initialises a "bad" route, it keeps going via this route with low probability to correct itself.

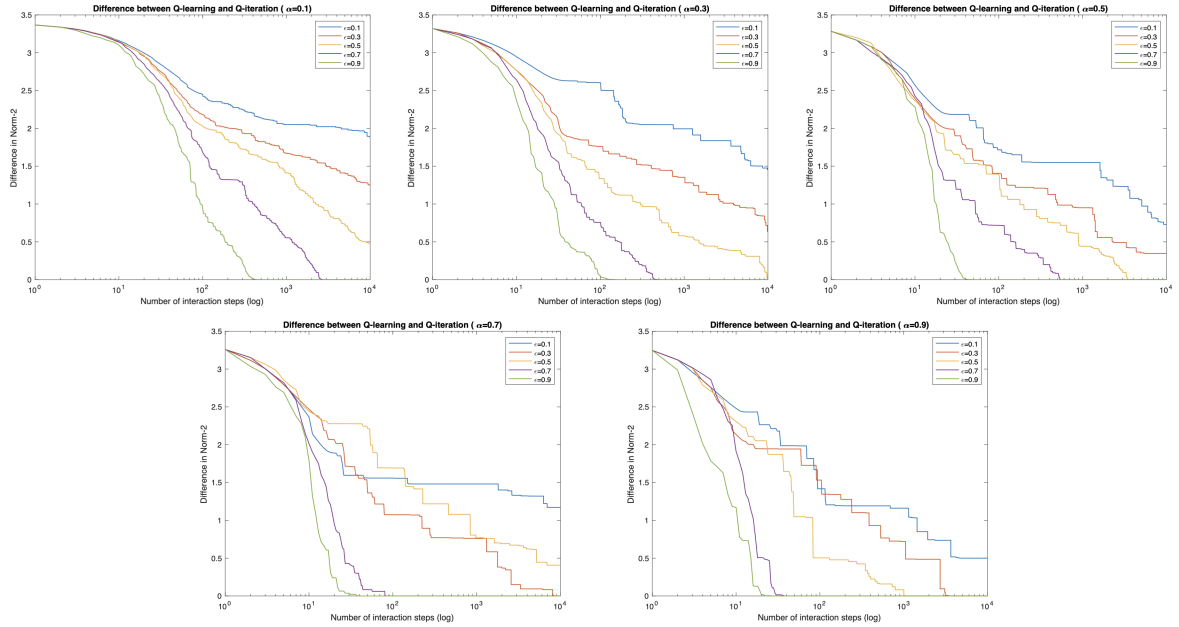


Figure 3: Differences between $Q$-learning and $Q$-iteration, fixed $\alpha$ with different $\epsilon$

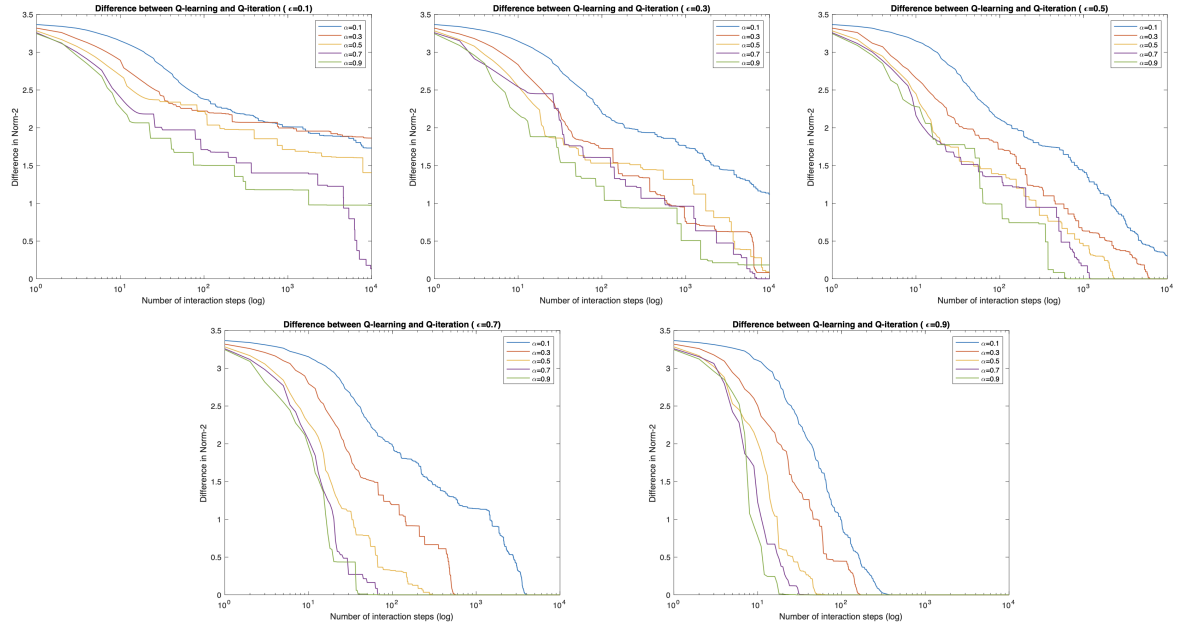


Figure 4: Differences between $Q$-learning and $Q$-iteration, fixed $\epsilon$ with different $\alpha$

# 4

Q-iteration is model-based method but Q-learning is model-free method. When executing Q-iteration, we already know the environment, say we have the model, each iteration the value function is updated globally until convergence, intuitively, agent knows the environment so just take the action which offer maximum reward. But for model -free method agent does not have information about this maze but have to learn, explore from the unseen environment, each execution step agent could only take one action locally. The difference is that whether the agent could predict the reward

and action for next step. I implement two method which could speed up Q-learning in certain conditions, the one is SARSA [2] and the other one is binary reward[1]. The iteration number is set 100000 repeated experiment number is 20.

For SARSA, it works well when $\alpha$ and $\epsilon$ are both small, the convergence speed of SARSA is faster than Q-learning, shown in Figure 5. Briefly, this method will decide action $a^{'}$ in state $s^{'}$, ones after initialisation in each episode next state and next action is calculated agent applies its own estimated actions as established policy which is know as on-policy, Q-learning is off-policy because each time the action is redetermined.

For binary reward, as shown in Figure 6 (the wide blue line high-lighted is Q-learning result as baseline), in this experiment, when initialisation value is smaller than 0.02, this method offers faster convergence than Q-learning while the value is larger than 0.05 the result is on the contrary, even worse than Q-learning. The idea behind this method, briefly, is that setting a positive small Q-value, after each iteration, the value of chosen action (direction) in a certain state would decrease, meaning next step this action is less possible to be chosen, so it is a method which could encourage agent to explore more rather than where they already know.

For the combination of these two methods, the result is shown in 7 ($\alpha, \epsilon = 0.1$, $Q$-initialisation=0.0). The result illustrates that the combination of these two methods could works faster than plain Q-learning method. Only thing to say is that this modification performs better holding on certain existed conditions but not for any condition.

The advantage of SARSA is that it works well when both $\epsilon$ and $\alpha$ are small in this situation function converges fast, the computational intensity is low, but as the disadvantage, when the parameter is large, SARSA performs rather bad the convergence steps required is way more larger than Q-learning. The advantage of binary reward is that the implementation is rather simple and for some certain situation the performance is rather good, but the disadvantage is that there is not a certain threshold value to determine which initialisation should be optimal in principle.

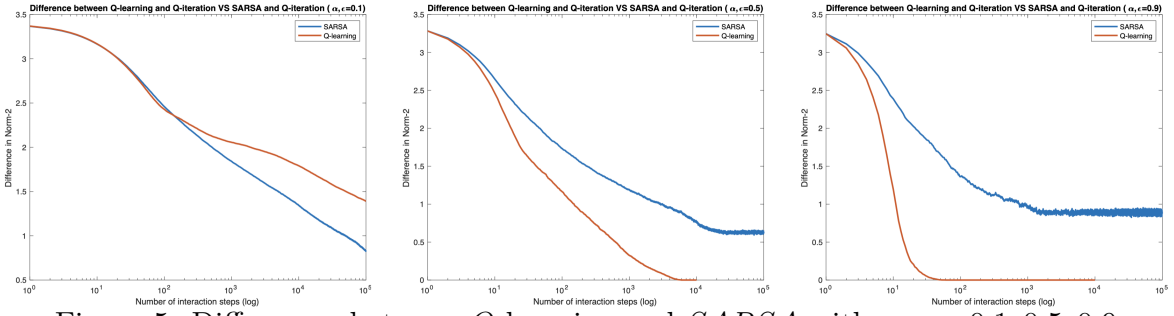


Figure 5: Differences between $Q$-learning and $SARSA$ with $\alpha, \epsilon = 0.1, 0.5, 0.9$

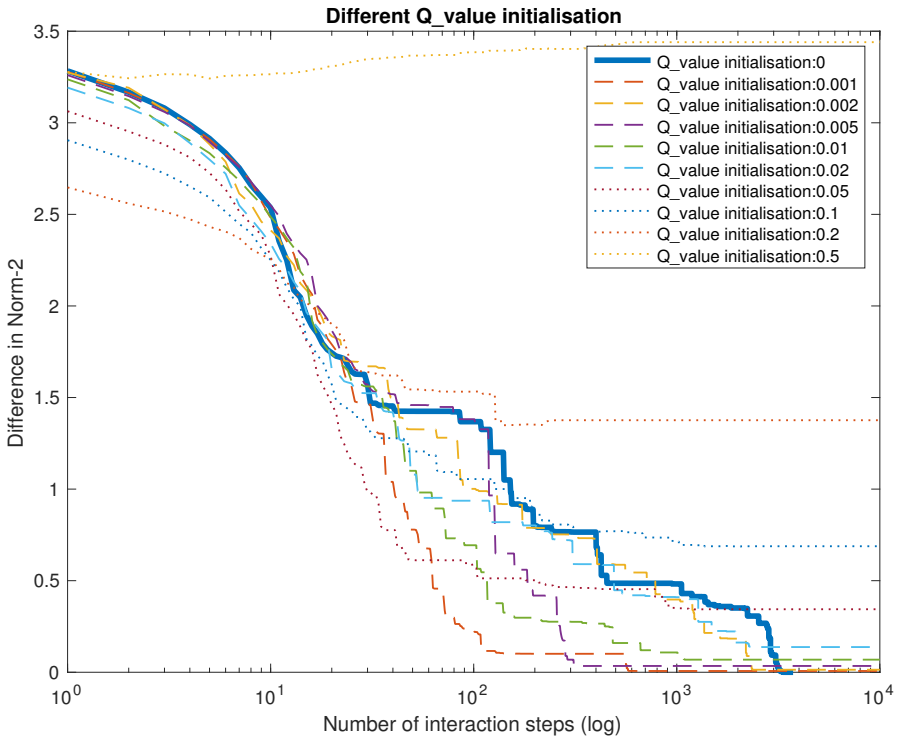


Figure 6: Differences between $Q$-learning with better initialisation and $Q$-iteration

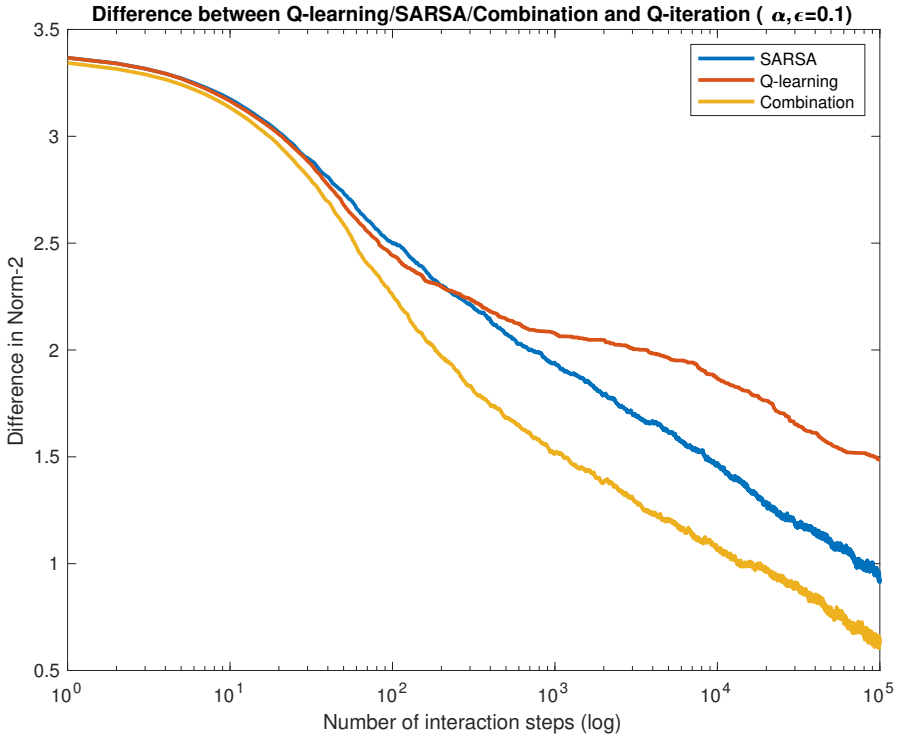


Figure 7: Differences between $Q$-learning with better initialisation/SARSA/Methods combination and $Q$-iteration ($\alpha, \epsilon = 0.1$, initialisation=0.001)

# Bibliography

[1] Laëtitia Matignon, Guillaume J. Laurent, and Nadine Le Fort-Piat. Reward function and initial values: Better choices for accelerated goal-directed reinforcement learning. In Stefanos D. Kollias, Andreas Stafylopatis, Włodzisław Duch, and Erkki Oja, editors, *Artificial Neural Networks – ICANN 2006*, pages 840–849, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[2] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009.