(CS 533) Intelligent and Decision Making

Homework 3:

Distributed Q-Learning and SARSA

A-young Kang (Student ID: 933-610-350)

1. **Did you observe differences for SARSA when using the two different learning rates? If there were significant differences, what were they and how can you explain them?**

   The results were different based on the learning rate. The learning rate determines how much newly acquired information overrides previous information. If the learning rate factor is zero, the agent will learn nothing. Otherwise, if the learning rate is 1, it would make the agent consider only the most recent information. As we can see, both right images from the figures, where the learning rate is 0.1, the average of total rewards fluctuate a lot throughout the episodes. On the other hand, both left images from the figures, where the learning rate is 0.001 which is much smaller than previous learning rate, the average of total rewards tends to fluctuate a lot at the early episodes and tends to converge at the later episodes.

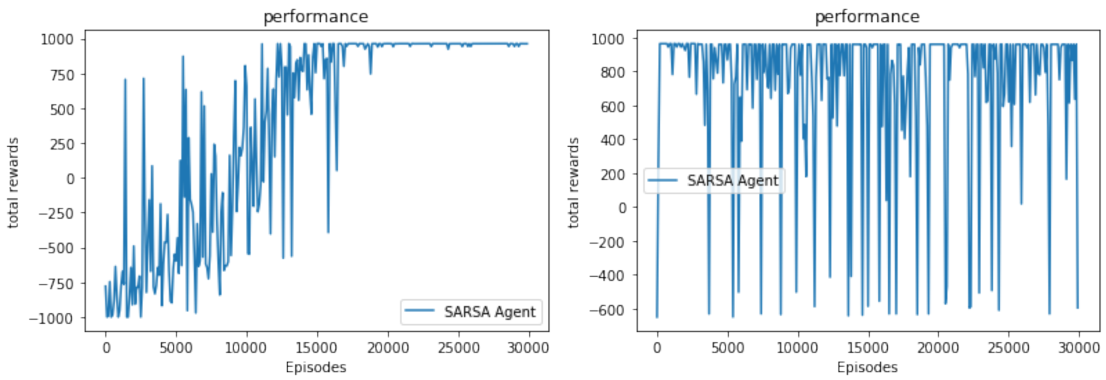


Figure 1. Learning curves for DH map using SARSA (L) α = 0.001, ϵ = 0.3 (R) α = 0.1, ϵ = 0.3

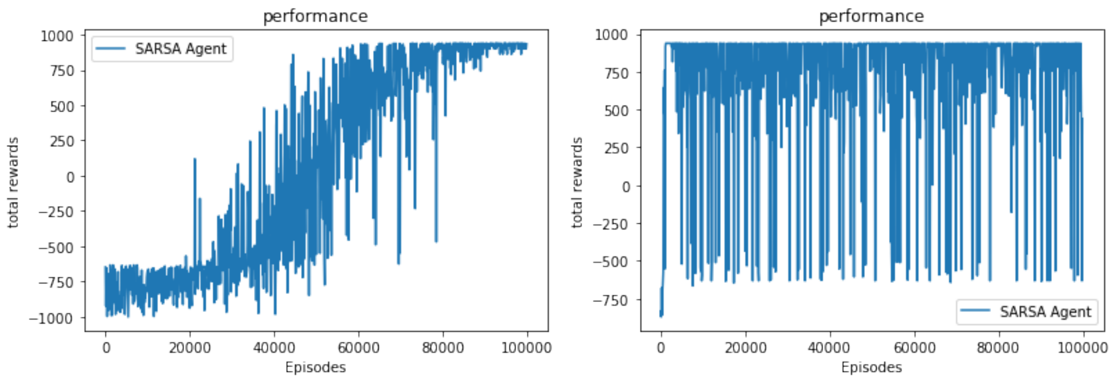


Figure 2. Learning curves for 16x16 map using SARSA (L) α = 0.001, ϵ = 0.3 (R) α = 0.1, ϵ = 0.3

2. **Repeat (1) for Q-Learning.**

   The results were different based on the learning rate. The learning rate determines how much newly acquired information overrides previous information. If the learning rate factor is zero, the agent will learn nothing. Otherwise, if the learning rate is 1, it would make the agent consider only the most recent information. Both left images in the figures, where the learning rate is 0.001, the total average rewards tends to fluctuate a lot at the early episodes and converges as the episodes get higher. Whereas, both right images in the figure, with the bigger learning rate, total average rewards converge little faster. However, at some higher episodes there were possibilities that total average rewards get smaller.

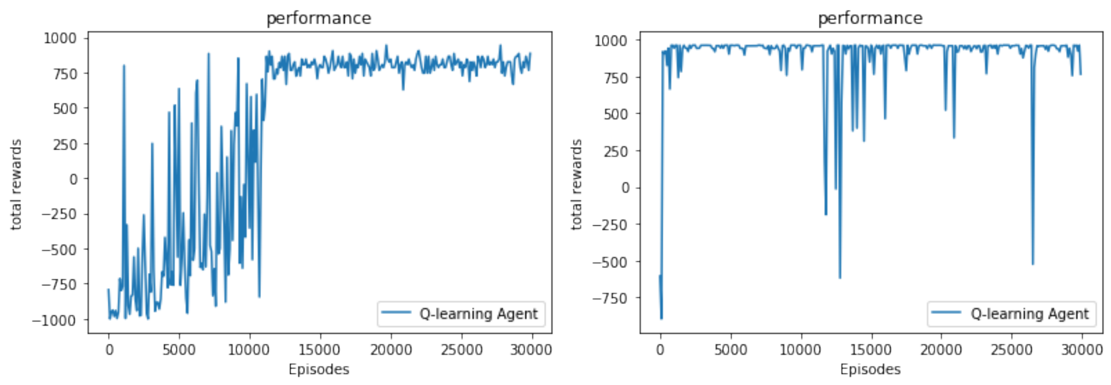


Figure 3. Learning curves for DH map using Q-Learning (L) α = 0.001, ϵ = 0.3 (R) α = 0.1, ϵ = 0.3

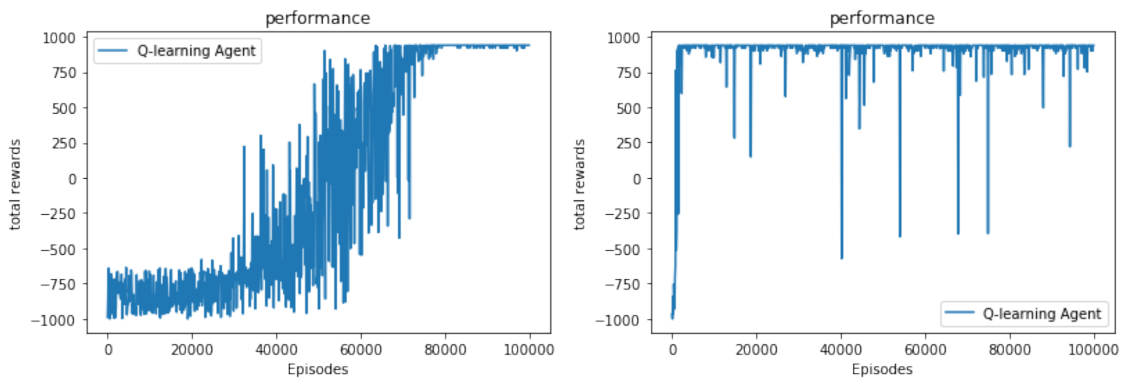


Figure 4. Learning curves for 16x16 map using Q-Learning (L) α = 0.001, ϵ = 0.3 (R) α = 0.1, ϵ = 0.3

3. **Did you observe differences for SARSA when using different values of ϵ? If there were significant differences, what were they and how do you explain them?**

   The results were different based on epsilon value. When the epsilon value is zero, the policy will pick the exploitation which is only based on the knowledge we know, whereas, when the epsilon value is 1, it will explore only disregarding any knowledge it gains. Both left images in the figures, where the epsilon value is 0.3, since it has bigger epsilon value than 0.05, it will do more exploration so we can see more oscillation and it converges in the higher episodes. However, in the right images, where the epsilon value is 0.05, the total rewards converges much faster than the former. This is because, the smaller the epsilon value, it will exploit more rather than explore, so the convergence appears much faster.

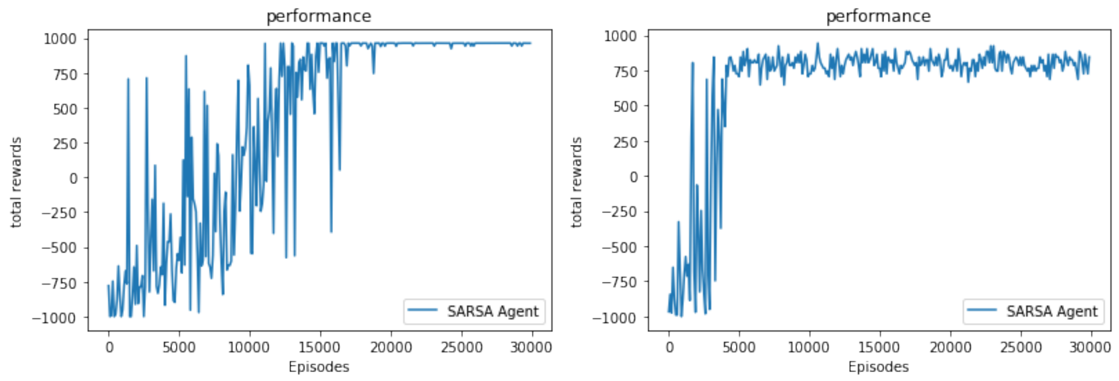


Figure 5. Learning curves for DH map using SARSA (L) α = 0.001, ϵ = 0.3 (R) α = 0.001, ϵ = 0.05

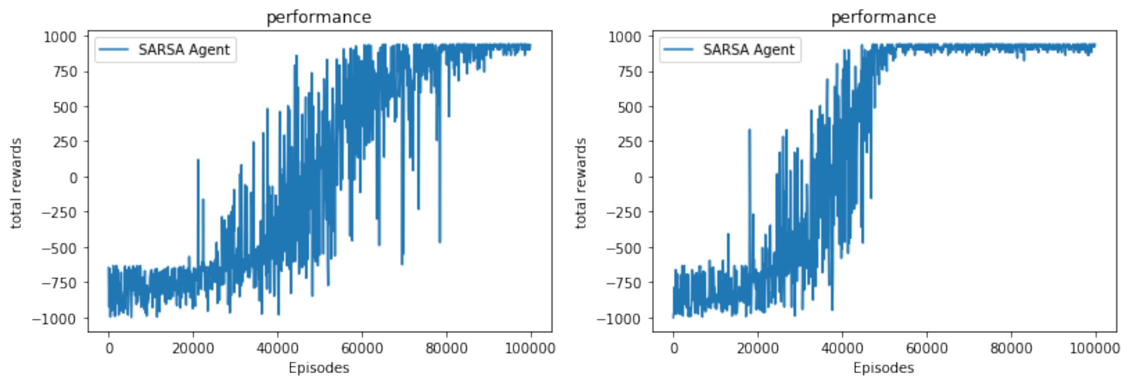
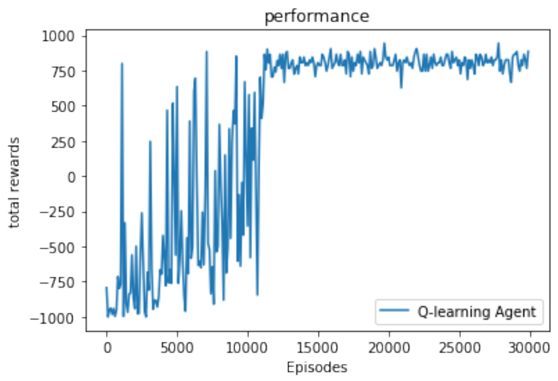


Figure 6. Learning curves for 16x16 map using SARSA (L) α = 0.001, ϵ = 0.3 (R) α = 0.001, ϵ = 0.05

4. **Repeat (3) for Q-Learning.**

   The results were different based on epsilon value. When the epsilon value is zero, the policy will pick the exploitation which is only based on the knowledge we know, whereas, when the epsilon value is 1, it will explore only disregarding any knowledge it gains. Both images in the left side, with the higher epsilon value 0.3, the policy goes for more exploration and the total rewards tend to fluctuate more and converges later. On the other hand, both right images, where the epsilon value is 0.05, the total rewards tend to converge faster. This is because with the smaller epsilon value, they exploit more and picks greedy policy during learning process.



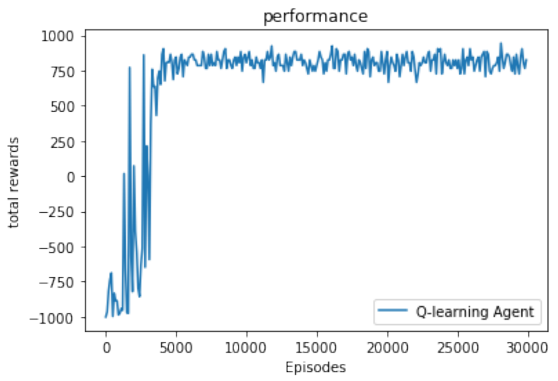
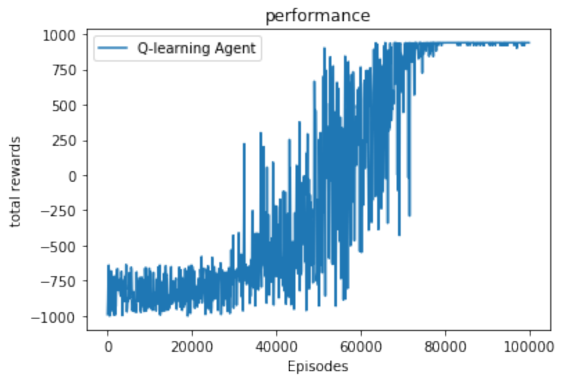


Figure 7. Learning curves for DH map using Q-Learning (L) α = 0.001, ϵ = 0.3 (R) α = 0.001, ϵ = 0.05



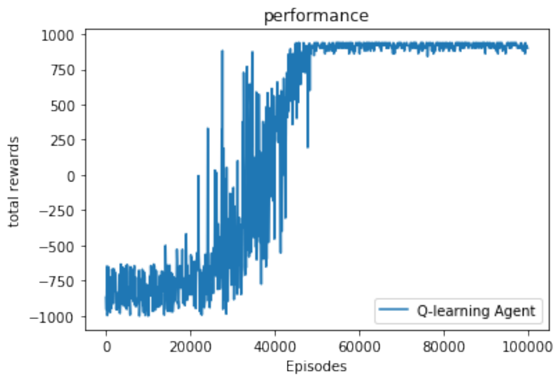


Figure 8. Learning curves for 16x16 map using Q-Learning (L) α = 0.001, ϵ = 0.3 (R) α = 0.001, ϵ = 0.05

5. **For the map "Dangerous Hallway" did you observe differences in the policies learned by SARSA and Q-Learning for the two values of epsilon (there should be differences between Q-learning and SARSA for at least one value)? If you observed a difference, give your best explanation for why Q-learning and SARSA found different solutions.**

   When SARSA is exploiting, it picks *a′* that maximizes the current Q-function, and thus it does sort of the same update as Q-Learning. However, they really differ on exploring: SARSA selects an action taken by the policy, while Q-Learning does a max over the actions at state *s′*. Therefore, when ϵ is equal to 0.3, the value functions by the two algorithms are quite different (Figure 9) because SARSA takes an action that may go against the current greedy action when exploring. The policies also look kind of different. However, when ϵ is equal to 0.05, both algorithms have only 5% chance of exploring and most of the time do exploitation, so the policies learned by SARSA and Q-Learning are similar (Figure 10).

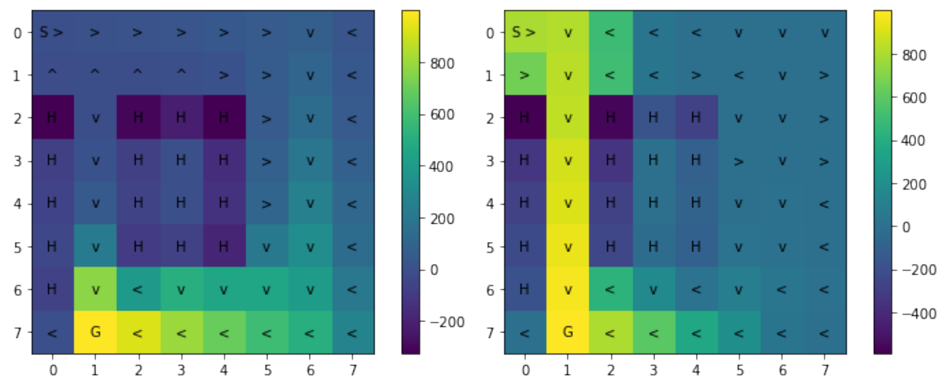


Figure 9. Policies learned by (L) SARSA with $\epsilon = 0.3$ (R) Q-Learning with $\epsilon = 0.3$

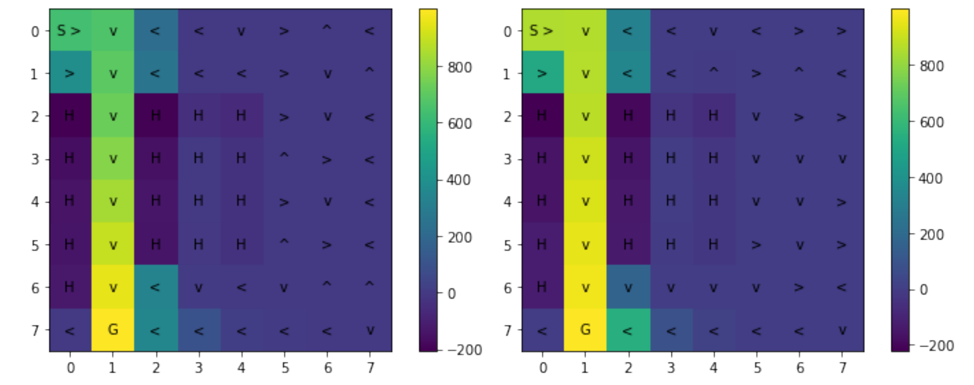
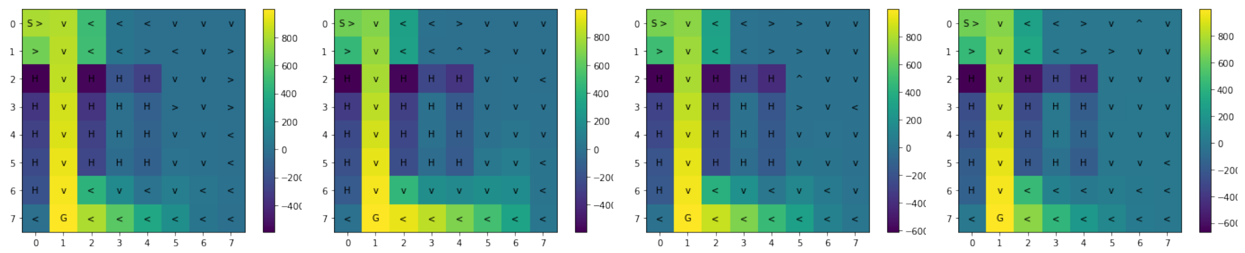


Figure 10. Policies learned by (L) SARSA with $\epsilon = 0.05$ (R) Q-Learning with $\epsilon = 0.05$

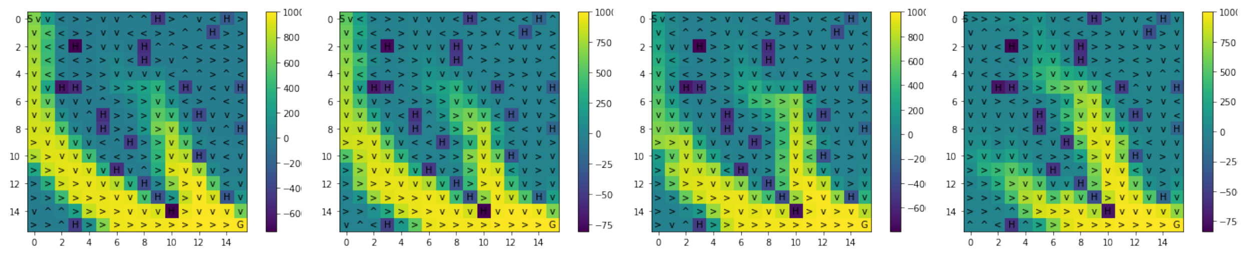
6. **Show the value functions learned by the distributed methods for the best policies learned with ϵ equal to 0.3 and compare to those of the single-core method. Run the algorithm for the recommended number of episodes for each of the maps. Did the approaches produce similar results?**

   Figure 11 describes value functions for the "Dangerous Hallway map" learned by the single-core method (a) and by the distributed methods (b, c, d). Figure 12 describes value functions for the "Size 16 map" learned by the single-core method (a) and by the distributed methods (b, c, d). As you can see, the non-distributed and distributed approaches produce similar value functions for the both maps.



(a) (b) (c) (d)

Figure 11. Value functions for DH map with ϵ = 0.3 (a) single core (b) cw = 2 (c) cw = 4 (d) cw = 8



(a) (b) (c) (d)

Figure 12. Value functions for 16x16 map with ϵ = 0.3 (a) single core (b) cw = 2 (c) cw = 4 (d) cw = 8

7. **Provide and compare the timing results for the single-core and distributed experiments, including the time to do the evaluations during learning. Describe the trends you observe as the number of workers increases.**

   Table 1 shows the time elapsed for the single-core and distributed approaches. When using the single-core method, it takes longer to produce a result than when using the distributed method for the both maps. As the number of collection workers increases from 2 to 4, the time taken becomes shorter but when we have 8 collection workers, the time elapsed increases. Since we always have the same number of evaluation workers which is 4, just to increase the number of collection workers would not contribute to better performance. Rather, it would take time to create the workers and wait for the workers to complete their jobs.

Method	**Dangerous Hallway Map**	**16x16 Map**
Single core	77.02963399887085	646.0700252056122
Collection workers = 2	20.056010484695435	89.6301941871643
Collection workers = 4	17.486527919769287	68.51794385910034
Collection workers = 8	19.064069986343384	76.26069855690002

Table 1. Time elapsed (sec) for the single-core Q-Learning and distributed Q-Learning approaches with the evaluation procedure turned on with α = 0.001, ϵ = 0.3

8. **Provide and compare the timing results for the single-core and distributed experiments with the evaluation procedure turned off. That is, here you only want to provide timing results for the learning process without any interleaved evaluation. Compare the results with (7).**

   Table 2 shows the time elapsed for the single-core and distributed approaches with the evaluation procedure turned off. Since we don’t include the evaluation procedure, the time taken is significantly reduced for the both methods. The overall time trends are similar as the number of collection workers increases compared to the results in Question (7).

Method	**Dangerous Hallway Map**	**16x16 Map**
Single core	12.48886513710022	121.26484394073486
Collection workers = 2	10.864487648010254	83.30143213272095
Collection workers = 4	9.147371053695679	59.870702028274536
Collection workers = 8	10.240160703659058	62.28326416015625

Table 2. Time elapsed (sec) for the single-core Q-Learning and distributed Q-Learning approaches with the evaluation procedure turned off with α = 0.001, ϵ = 0.3