CS 533: Intelligent Agent and Decision
Homework 3

Report by:

Christopher Buss
Sanad Saha

1. Provide the learning curves for the above experiments. Clearly label the curves by the parameters used.

Answer:
The following learning curves are for the **Q-Learning agent** on the **dangerous hallway map**. We set epsilon = 0.3 (default) and tested it for learning rate 0.1 and 0.001.

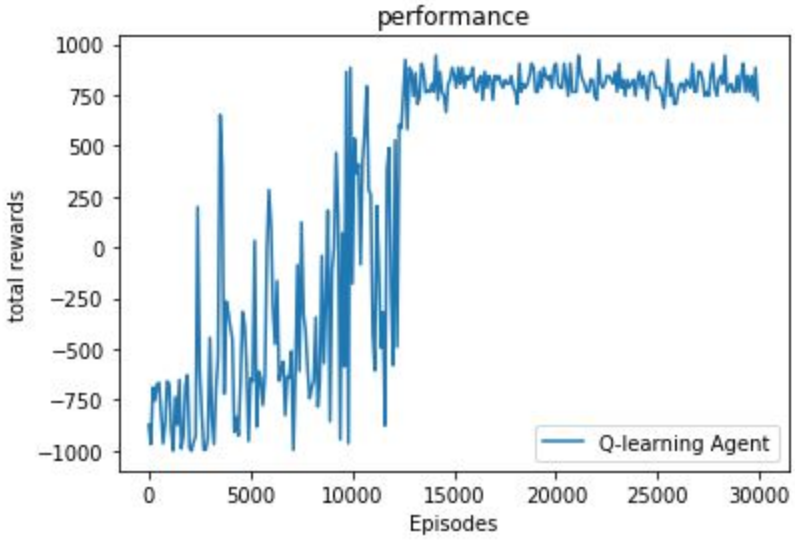

Figure 1: Q-Learning Agent Learning Curve for epsilon 0.3, MAP = map_DH and Learning rate = 0.001

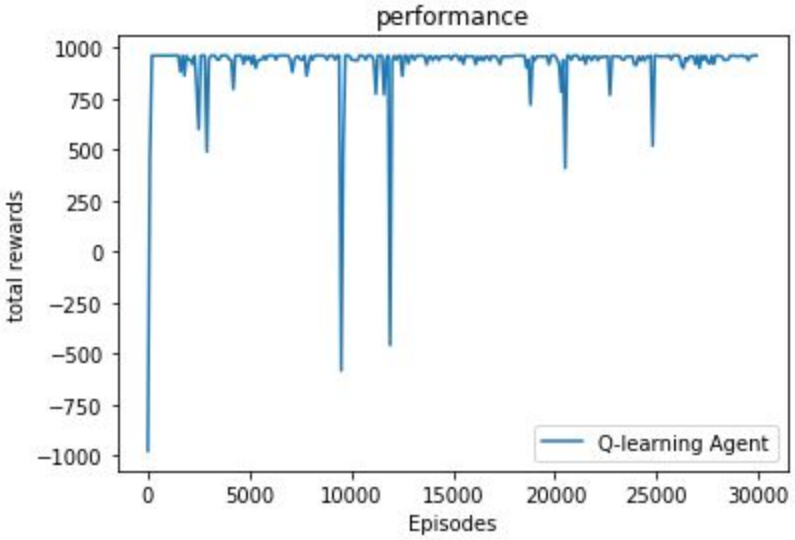

Figure 2: Q-Learning Agent Learning Curve for epsilon 0.3, MAP = map_DH and Learning rate = 0.1

We saw that the agent with learning rate 0.001 was performing much better than the agent with learning rate 0.1. So used learning rate 0.001 and tested on the dangerous hallway epsilon 0.05

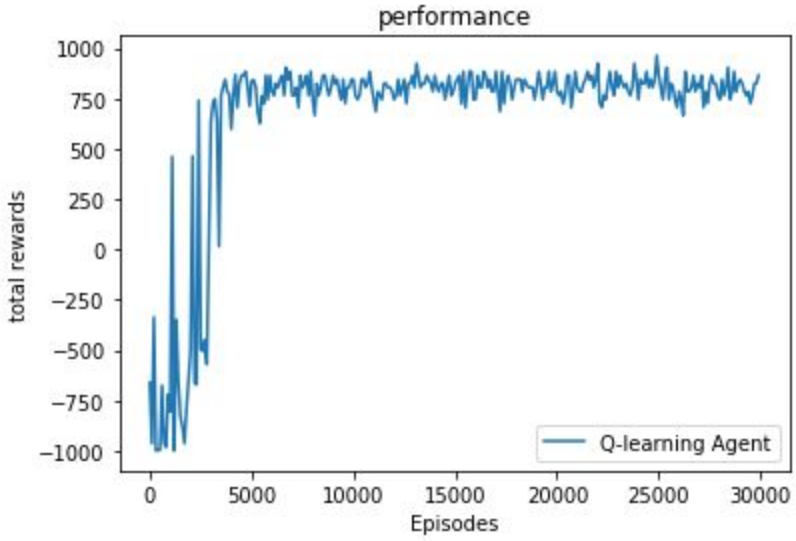

Figure 3: Q-Learning Agent Learning Curve for epsilon 0.05, MAP = map_DH and Learning rate = 0.001

The following learning curves are for the **Q-Learning agent** on the **16*16, map_16**. We set epsilon = 0.3 (default) and tested it for learning rate 0.1 and 0.001.

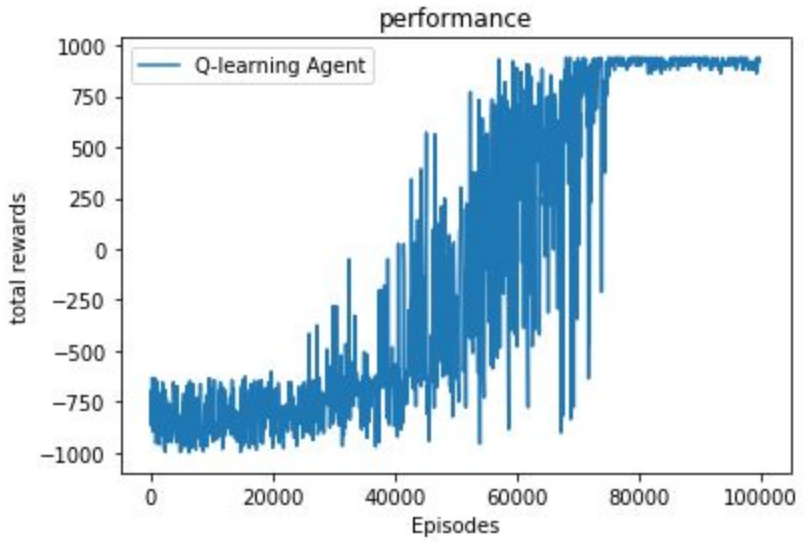

Figure 4: Q-Learning Agent Learning Curve for epsilon 0.3, MAP = map_16 and Learning rate = 0.001

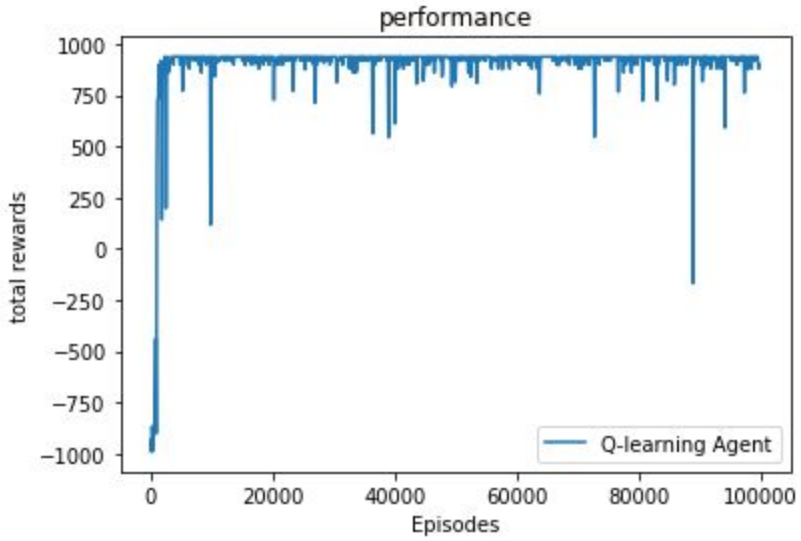

Figure 5: Q-Learning Agent Learning Curve for epsilon 0.3, MAP = map_16 and Learning rate = 0.1

We saw that the agent with learning rate 0.001 was performing much better than the agent with learning rate 0.1. So used learning rate 0.001 and tested on the map_16 with epsilon 0.05

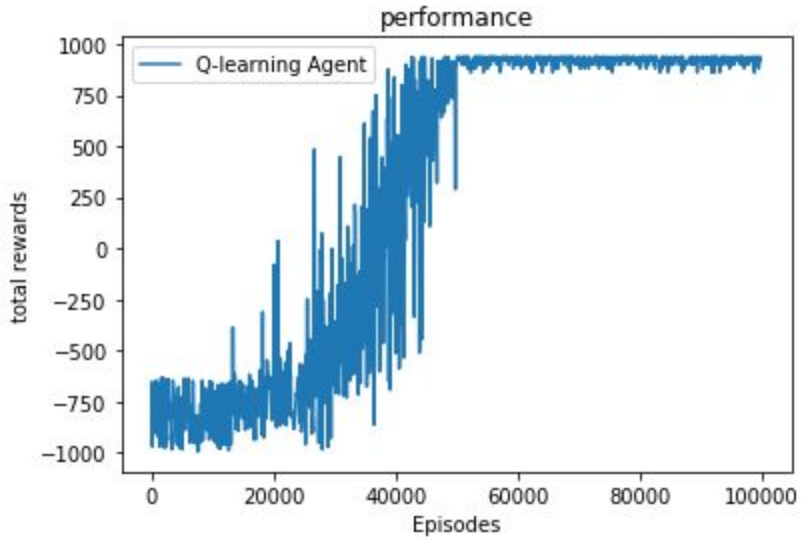

Figure 6: Q-Learning Agent Learning Curve for epsilon 0.05, MAP = map_16 and Learning rate = 0.001

The following learning curves are for the **SARSA agent** on the **dangerous hallway map**. We set epsilon = 0.3 (default) and tested it for learning rate 0.1 and 0.001.

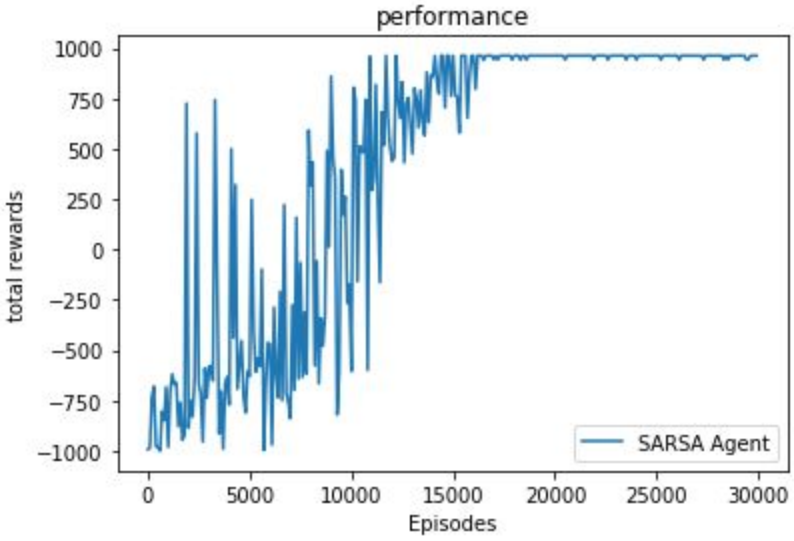

Figure 7: SARSA Agent Learning Curve for epsilon 0.3, MAP = map_DH and Learning rate = 0.001

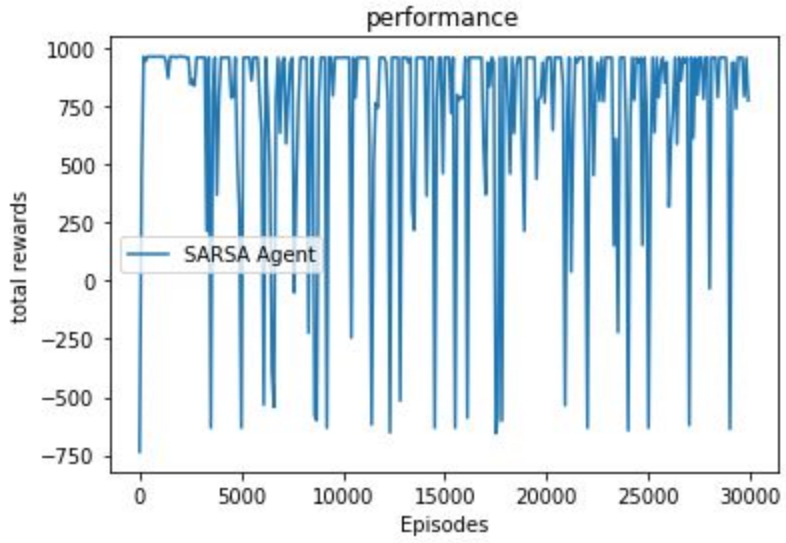

Figure 8: SARSA Agent Learning Curve for epsilon 0.3, MAP = map_DH and Learning rate = 0.1

We saw that the agent with learning rate 0.001 was performing much better than the agent with learning rate 0.1. So used learning rate 0.001 and tested on the dangerous hallway epsilon 0.05

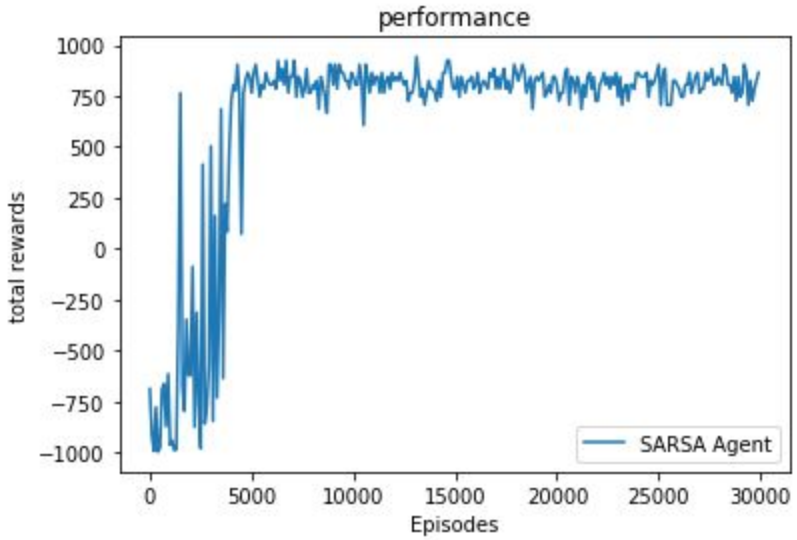

Figure 9: SARSA Agent Learning Curve for epsilon 0.05, MAP = map_DH and Learning rate = 0.001

The following learning curves are for the **SARSA** on the **16*16, map_16**. We set epsilon = 0.3 (default) and tested it for learning rate 0.1 and 0.001.

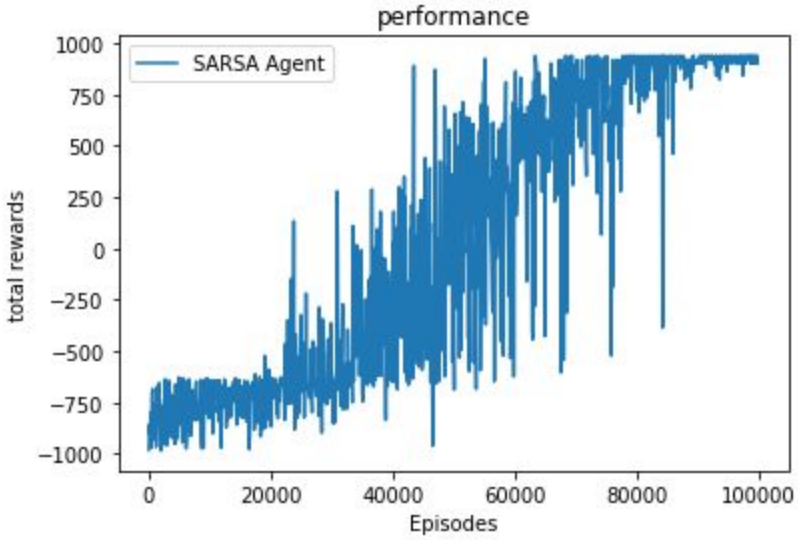

Figure 10: SARSA Agent Learning Curve for epsilon 0.3, MAP = map_16 and Learning rate = 0.001

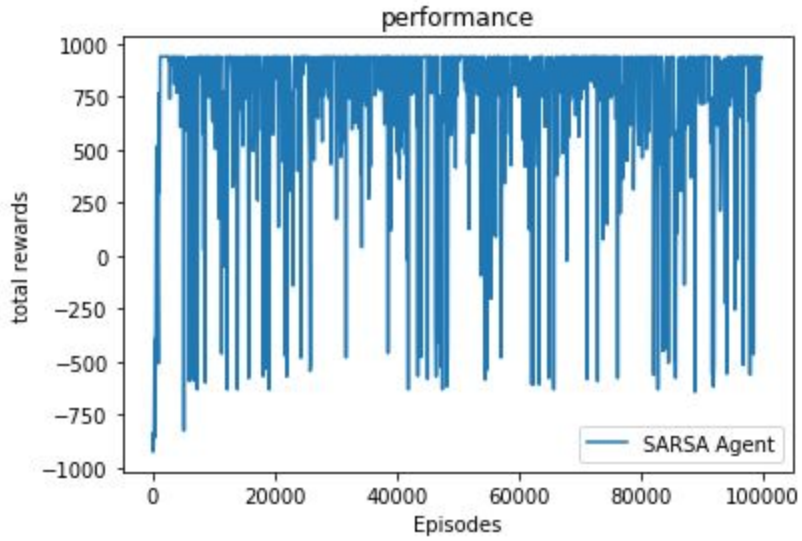

Figure 11: SARSA Agent Learning Curve for epsilon 0.3, MAP = map_16 and Learning rate = 0.1

We saw that the agent with learning rate 0.001 was performing much better than the agent with learning rate 0.1. So used learning rate 0.001 and tested on the map_16 with epsilon 0.05

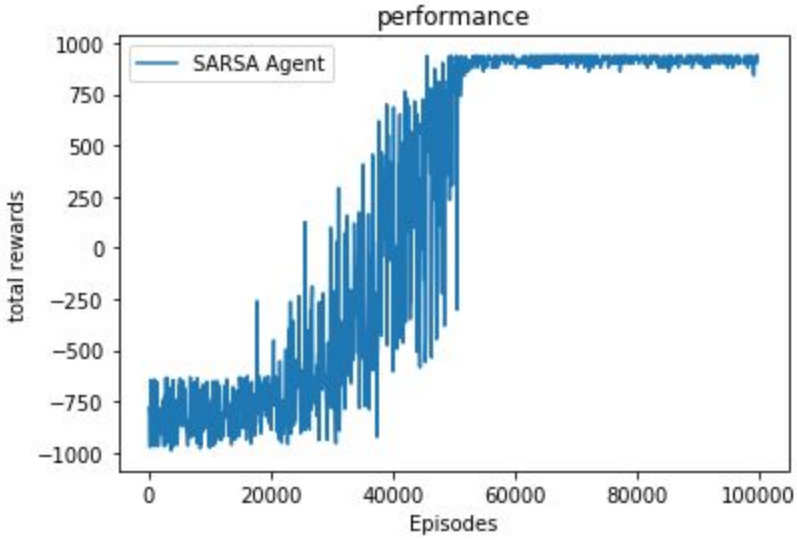

Figure 12: SARSA Agent Learning Curve for epsilon 0.05, MAP = map_16 and Learning rate = 0.001

2. 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?

Answer:

Yes. There are definitely some differences for SARSA when we are using two different learning rates. In both map_16 and map_DH (Dangerous Hallway) using a smaller learning rate (0.001) is better than using the larger one.

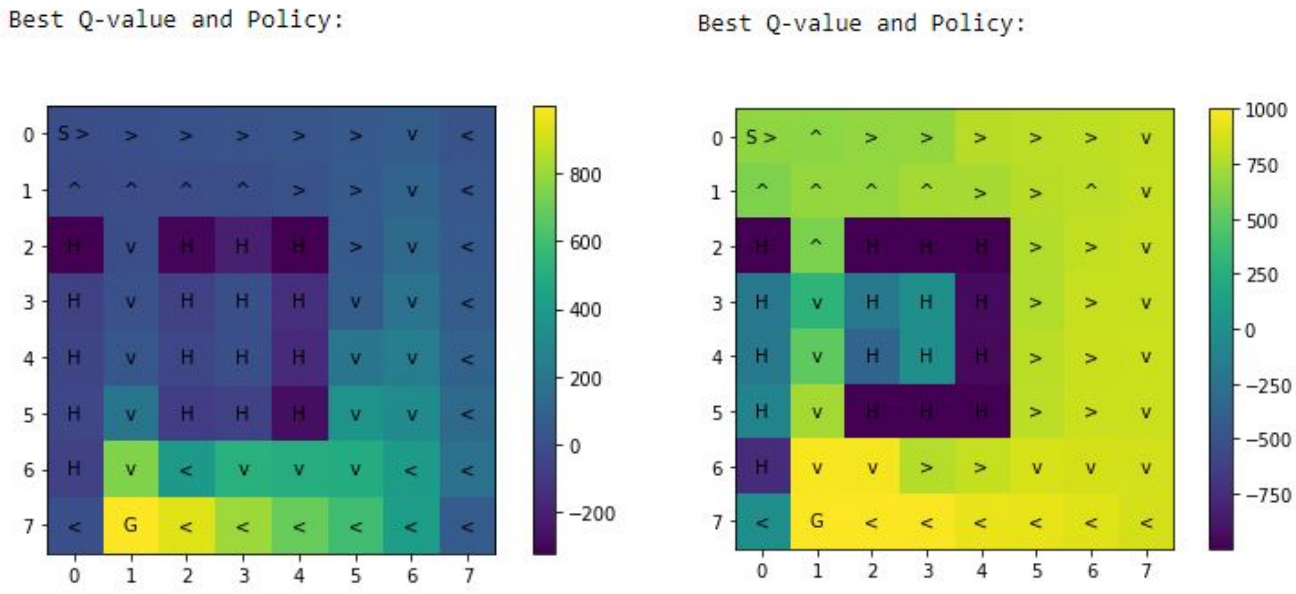

Figure 13: Policy comparison in the dangerous hallway for SARSA agent Left one for learning rate 0.001 and the right one for learning rate 0.1

A larger learning rate makes the values of the states decay less as they are propagated backwards, this is why more states in the right graph show higher values. This looks as though it makes the Q-function noisier as the values of the states grow closer to the max reward.

3. Repeat (2) for Q-Learning.

Answer:

Below are two graphs for Q-Learning with different learning rates.

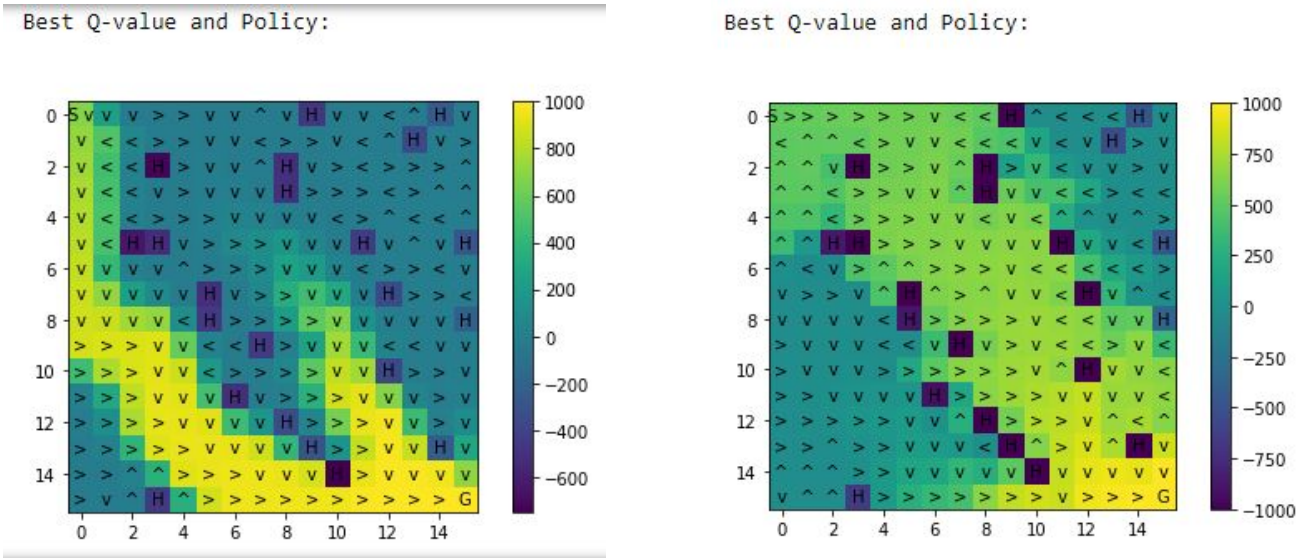

Figure 14: Policy comparison in the map_16 for Q-Learning agent. Left one for learning rate 0.001 and the right one for learning rate 0.1

As we have seen above, the lower learning rate produces less noise in the policy. In the left graph, we see that the agent's policy is more linear, with a clear path with high value. However, the graph on the left shows a much noisier policy.

4. 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?

Answer:

Below are two graphs showing different policies for different epsilon values.

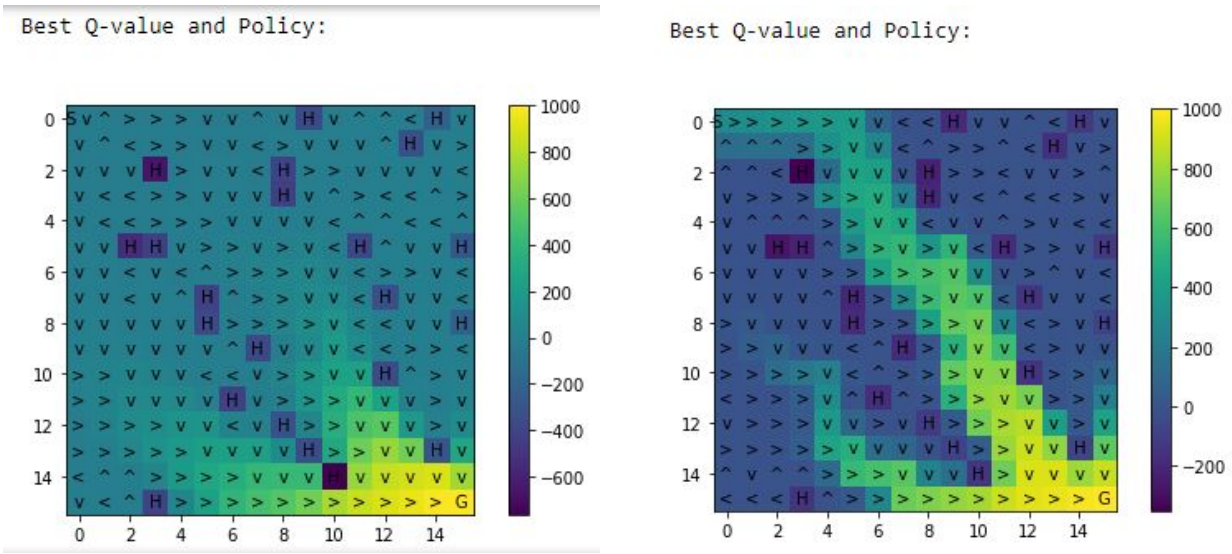

Figure 15: Policy comparison in the map_16 for SARSA agent. Left one for epsilon 0.3 and the right one for epsilon 0.05 [learning rate 0.001]

With an epsilon of .3, the SARSA agent takes a lot of random actions, meaning that it is exploring too much. Since it explores often, many states look worse than they actually are. With a lower epsilon, our agent still explores (notice the higher value states moving from the (9,1) to the goal), but does not do so to the detriment of its policy.

5. Repeat (4) for Q-Learning.

Answer:

Below are two graphs showing different policies for different epsilon values.

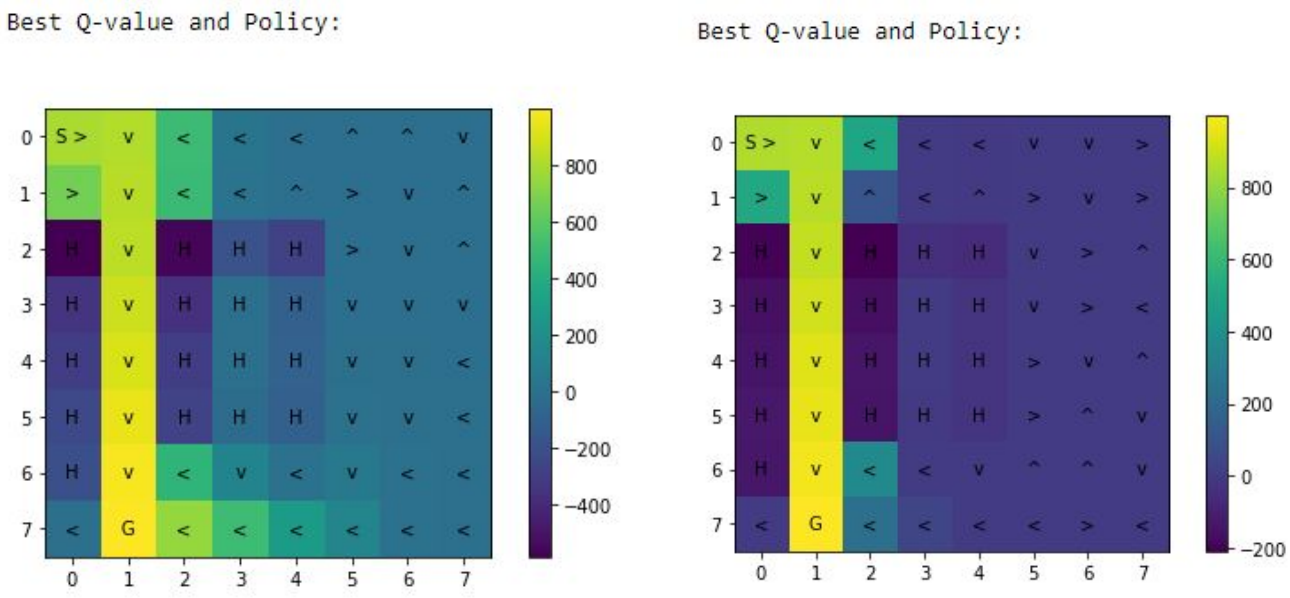

Figure 16: Policy comparison in the Dangerous Hallway map for Q-Learning agent. Left one for epsilon 0.3 and the right one for epsilon 0.05 [learning rate 0.001]

Notice how the left graph has higher values for almost all of its states, meaning that it explored quite a bit. But in the end, both agents found the same policy. Q-Learning seems to have a higher tolerance for exploration than SARSA in this case.

6. 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.

Answer:
Below are two graphs showing SARSA and Q-Learning agents policies on the Dangerous Hallway map with an epsilon of 0.3.

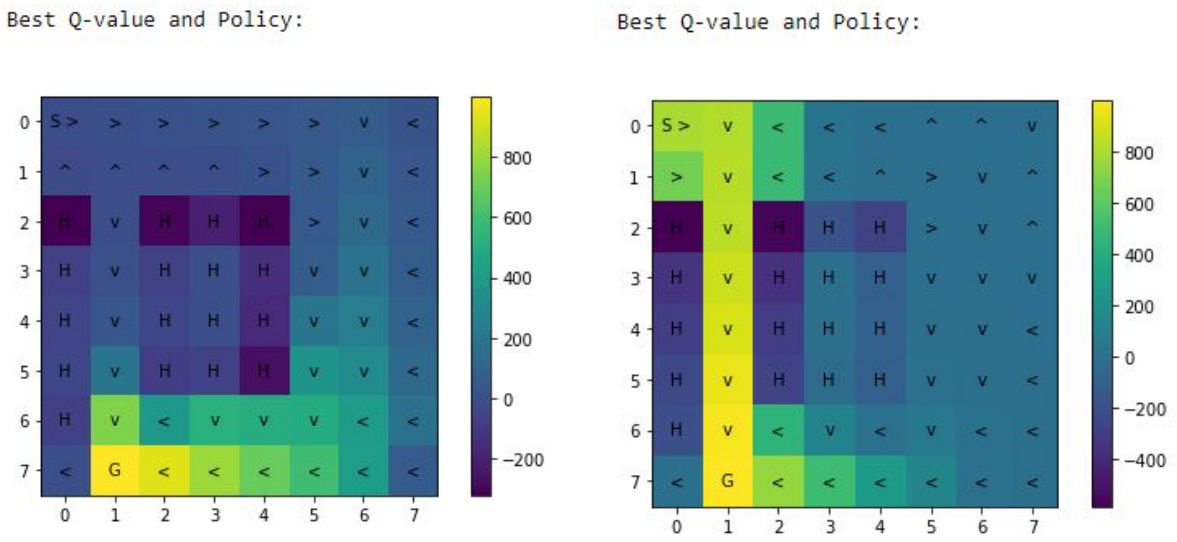

Figure 17: Policy comparison in the Dangerous Hallway map for SARSA[left] Q-Learning agent[Right]. [epsilon 0.3 , learning rate 0.001]

Both SARSA and Q-Learning agents look ahead a step to the next state (proceeding the one they just entered). The difference is that SARSA chooses that next state with an explore-exploit strategy whereas Q-Learning uses a greedy strategy. This means that bad actions made in the name of exploration have less of an effect on the learned policy. Another way to think about this is that SARSA is more sensitive to situations where an explore action could lead to a bad reward. This is why SARSA wants to move around the hallway rather than through it.

This changes when we turn epsilon down to 0.05, as we see below.

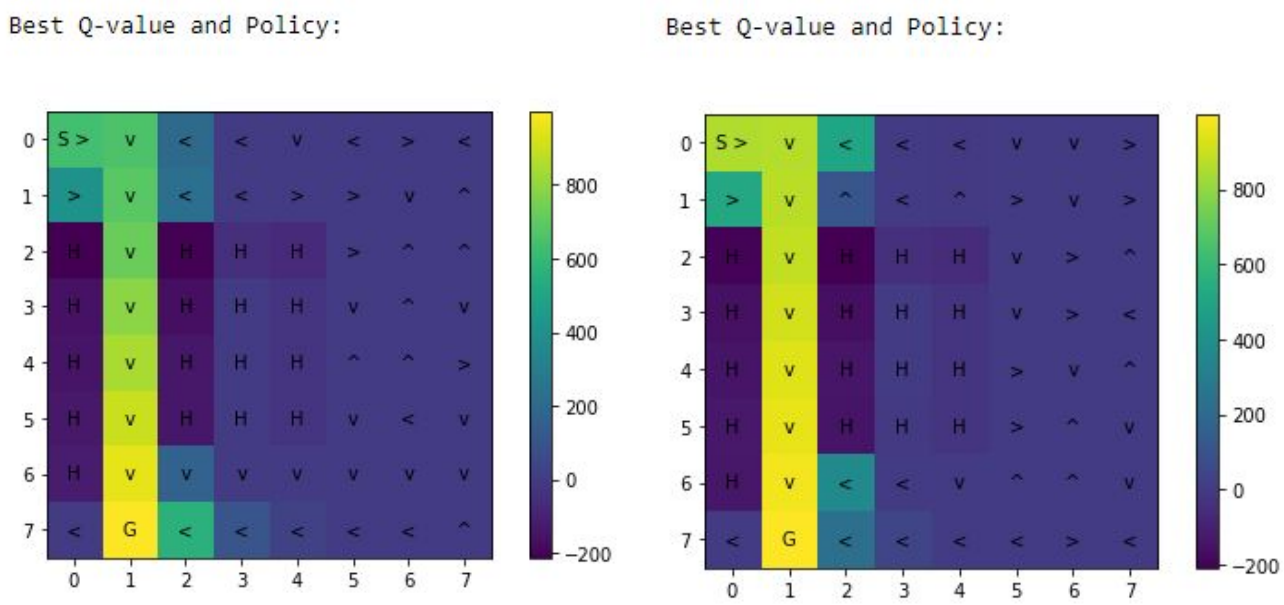

Figure 17: Policy comparison in the Dangerous Hallway map for SARSA[left] Q-Learning agent[Right]. [epsilon 0.05 , learning rate 0.001]

SARSA now follows the same policy as Q-Learning, but with less confidence. SARSA is still affected by its high exploration, but since the probability of exploration is significantly lower, it is less likely to choose bad actions that hurt it a lot when moving through the hallway.

7. 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?

Answer:

From the following figure 18 and 19 we can see that for both distributed Q-learning and non distributed Q learning approach the policies produce similar results. But the non-distributed version is more confident about its policy for both the maps from the beginning contrary to the distributed version.

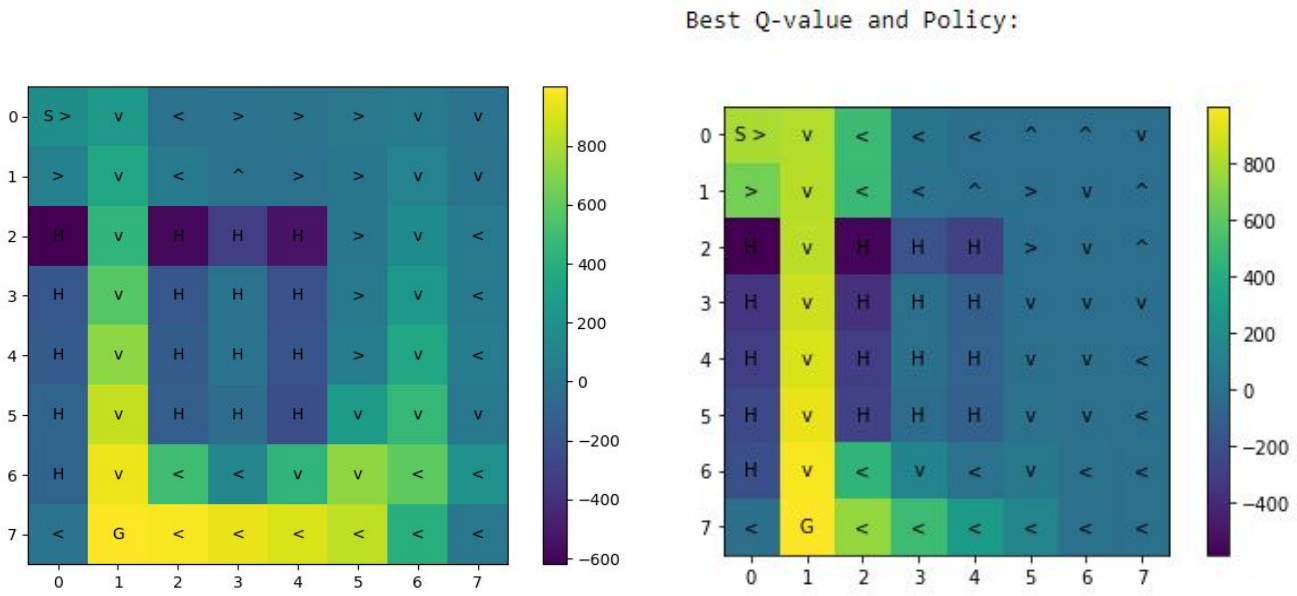

Figure 18: Policy comparison in the Dangerous Hallway map for SARSA for Distributed [Left] and Non-Distributed[Right] For Epsilon 0.3

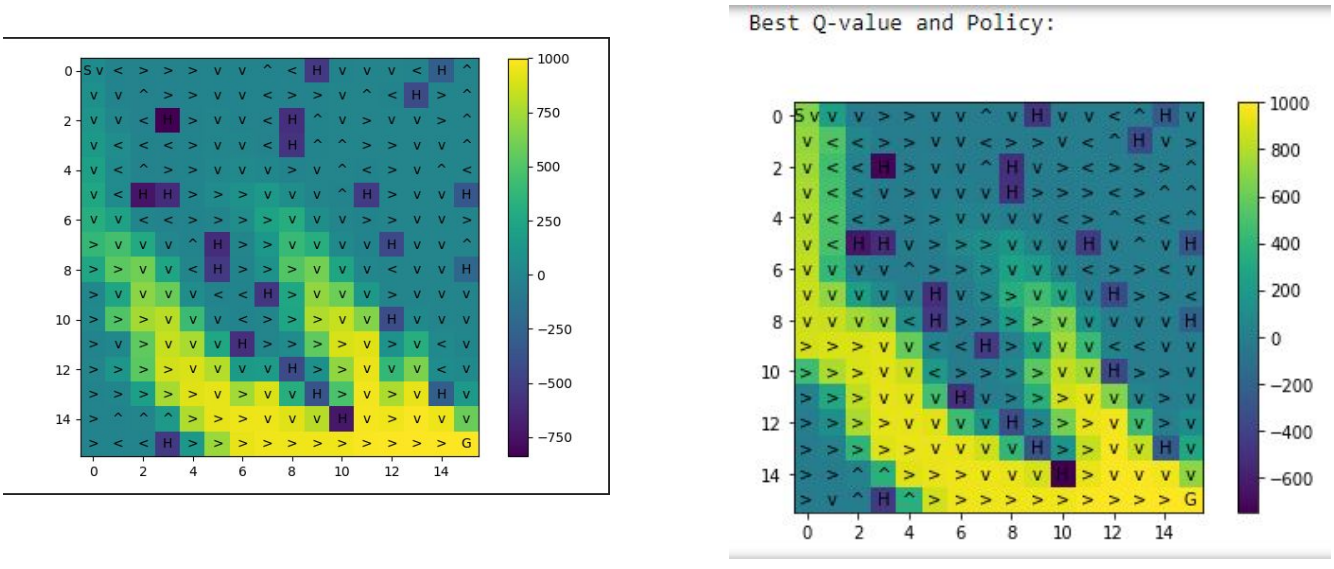

Figure 19: Policy Comparison in map_16 for SARSA for Distributed [Left] and Non-Distributed[Right] For Epsilon 0.3

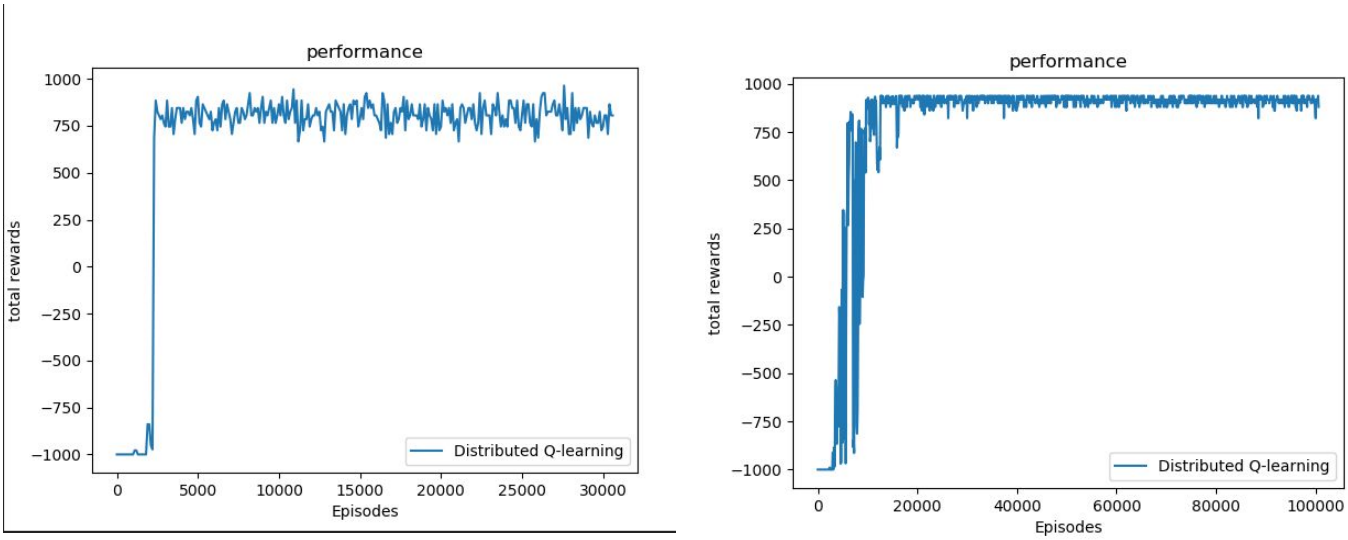

Figure 20: Learning curve for Dangerous hall map [left] and map_16 [map] for Q-Distributed learning using Epsilon 0.3 and Learning rate 0.001

We ran the Q-Learning algorithm on both of the maps and figure 20 provides us how quickly we reach convergence. If we see our learning curves from the question 1 we would find that Distributed version of the Q-Learning algorithm provides ridiculously fast convergence to the best policy. Because multiple workers are working simultaneously toward the same goal.

8. 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.

Answer:

Table 1 provides the comparison of timing results for the single-core and distributed experiments; evaluation was turned on during learning for both of the cases. We can also see that on both Dangerous Hall and map_16 the single-core Q-Learning performs poorly compared to the distributed version.

	Single-Core Q-Learning	Distributed Q-Learning
Dangerous Hall Map	49.67219829	8.697365
map_16	415.498228	49.502

Table 1: Comparison between timing results (in Seconds) while doing the evaluations during learning [epsilon = 0.3 and learning rate = 0.001]

We tried with multiple different combination of workers and the runtime gets reduced as expected when we use lot of workers. Although, from our experiments we don't expect timing to continuously improve as we increase the number of workers; which is visible from the column 2 and column 3. But at least it gets significantly better than using low number of workers.

Distributed Q-Learning	collector worker = 8 evaluation worker = 4	collector worker = 8 evaluation worker = 8	collector worker = 2 evaluation worker = 2
Dangerous Hall Map	8.697365	8.47	11.699
map_16	49.502	44.32	72.2137

Table 2: Comparison between timing results (in Seconds) while doing the evaluations during learning [epsilon = 0.3 and learning rate = 0.001]

9. 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 (8).

Answer:

From the following tables we can see that there is significant improvement (80% decrease in running time) in the Single Core Q-Learning when evaluation is turned off compared to when evaluation is turned on. [Table 3]

	Single-Core Q-Learning No Evaluation	Single-Core Q-Learning
Dangerous Hall Map	8.730796	49.67219829
map_16	70.0878	415.498228

Table 3: Comparison between timing results (in Seconds) with evaluation turned-off[left] and on [right] during learning [epsilon = 0.3 and learning rate = 0.001]

	Distributed Q-Learning	Distributed Q-Learning
Dangerous Hall Map	8.697365	7.38430266
map_16	49.502	36.824

Table 4: Comparison between timing results (in Seconds) with evaluation turned-on[left] and off [right] during learning [epsilon = 0.3 and learning rate = 0.001]

But the margin of improvement is not so significant in the Distributed case, although provided that distributed Q-Learning is already very fast in the first place; it can be said turning off evaluation while Distributed Q-Learning gives nice speed boost. [Table 4]

Not to mention, this performance measurement is a relative comparison rather than absolute. Which can be credited to the busyness of the intel devcloud which was giving different runtime during different runs. For each test we ran it more than 4 times and portrayed the timing with the minimum runtime.