Homework 5 - Reinforcement learning Learning

Alberto Zancanaro - ID: 1211199

I. Introduction

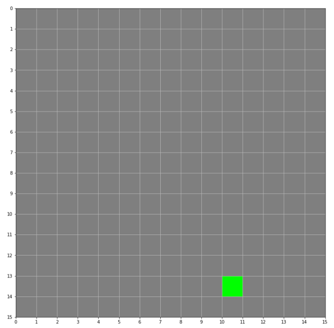
This homework was focused around the creation and testing of a Reinforcement Learning (RL) Model and evaluate its actions inside a virtual environment. We will test various configuration and learning behaviors and evaluate the consequent results.

Reinforcement Learning is a machine learning technique developed to encourage learning through a reward/penalty system in order to maximize the total reward at the end of the process. This technique allow an agent to learn how to navigate in a virtual environment subject to certain rule.

II. Problem Structure

The agent that implement the RL algorithm has the task of navigate through an environment to reach a goal point. The environment is an $x \times y$ grid and at each time step the agent can be choose between five action: stay or move in one of the four direction (up, down, left, right). No diagonal movement has been implemented. Also the environment can be modify by adding obstacles.

The agent will obtain a positive reward(+1) if he reach the goal point or a negative reward if he touch an obstacles(-2) or try to exit the border(-1). The agent implements Q-Learning whit greedy decision policy and SARSA Learning with both greedy and softmax decision policy.


(a) Environment without obstacle

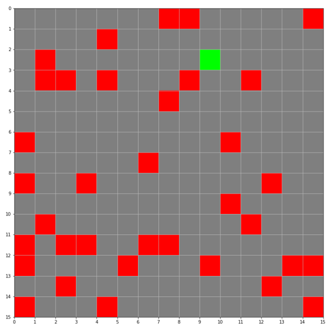

(b) Environment with obstacle

Fig. 1: Possible Environment

In figure 1 it is possible to observe two different environment of 15×15 square[1]. The green point is the goal point and the red points in the second figure are the obstacles. Both goal points and obstacles are randomly set inside the grid. The initial position of the agent is also randomly select. Cross-check ensures that there are no overlaps between obstacles, goal point and initial position.

[1]Since some problem with *matplotlib* the line of the grid and the square can be not perfectly aligned everywhere

The initial test were focused about correct code implementation, confrontation between learning policy and hyper-parameters tuning. After that more complex rule were added and tested (penalty to go near obstacles and multiple goal).

III. Code Implementation

All the model was implemented with *Python 3.7.6*; no python framework for machine learning was used during implementation. The *matplotlib* library was used as support for visualization.

IV. Results - Algorithm Comparison

All the following results are obtained with the subsequent general settings:

[x y]	[15 15]
Total Episodes	4000
Episode Length	66
Number of obstacles	35

The α tested were: $[0.1, 0.2, 0.4, 0.6, 0.8, 1]$.

The spawn point of the agent is always random and at least 4 square away[2] from the goal. This was done to ensure that the agent learns in every situation to follow a slightly longer path. The distance of 4 is related to the dimension of the environment since it is not big enough to prevent high score in final episodes but at the same time is big enough for path visualization.

The obstacles and goal can be randomly generate but for each test the same layout is used. In this way we comparison between learning methods are more accurate and robust.

A representation of the environment is visible in figure 2.

The various figure do not represent the value for each episodes since the high number of episodes would have made the figures confusing and difficult to read. Instead, for clarity of the exposure, was choose to take an average every n episodes (respectively $n = 40$ episodes for mean reward figures and $n = 30$ episodes for action/stay figure).

Since the legend of the various alpha can be too small to be read in the figure the blue line corresponds to $\alpha = 0.1$, the orange one to $\alpha = 0.2$, the green one to $\alpha = 0.4$, the red one to $\alpha = 0.6$, the violet one to $\alpha = 0.6$ and the brown one to $\alpha = 1$. This choice regarding the figure dimension was made to maintain the figure as close as possible to their relative part of report. In the mean reward figure was also added a pink line that represent at 1, the hypothetical maximum reward. Since the agent must spawn at least 4 square away from the goal the rewards 1 can not be achieved by any agents but the best agent will have a very near reward at the end of the training.

[2]The code allows to set any number.

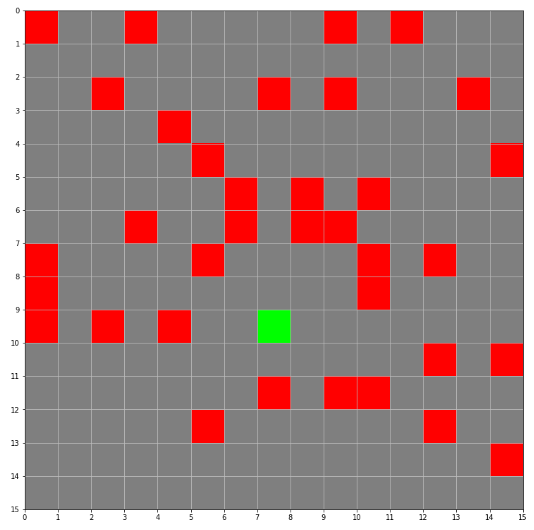
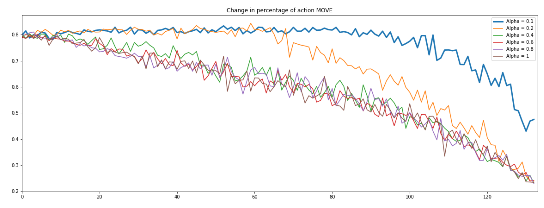
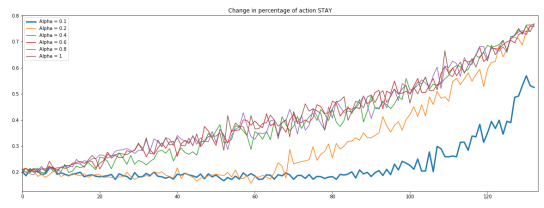


Fig. 2: Environment used for testing



(a) Percentage of action move



(b) Percentage of action stay

Fig. 3: Percentage of actions stay and move (Q-Learning)

A. *Q-Learning (Softmax and Greedy Policy)*

In this test we try the Q-learning algorithm with both a softmax decision policy and a ϵ greedy decision policy. The figure are the ones of the softmax case since in both case the results were very similar. So the analysis of both cases will be equal and all conclusion will be valid for both case.

In figure 3 is possible to observe the percentage of action stay and move. We can noticed that increasing alpha (α) the percentage of action move decrease more quicker and the percentage of action stay increase more quicker. We can also note that as time progress the percentage of move action decrease and stay action increase. This is coherent with a learning process since more the agent learn more quick must be its path to the goal.

From figure 4 we can see that the mean reward also increase

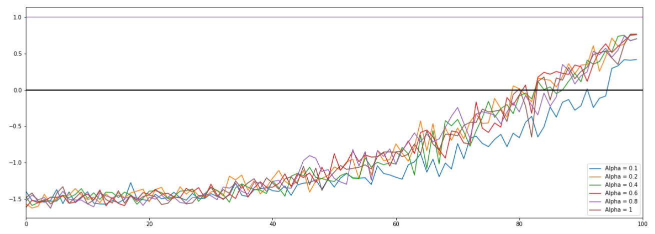


Fig. 4: Mean rewards (Q-Learning)

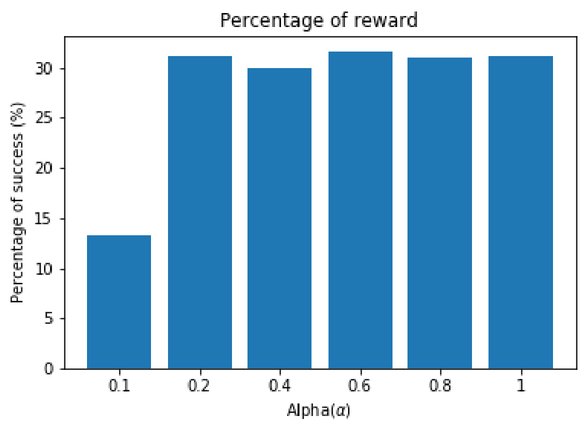


Fig. 5: Percentage of episodes with positive reward (Q-Learning)

slightly quicker with a bigger alpha[3]. This is coherent with our previous analysis of the the percentage of stay/move action.

In figure 5 we can see the percentage of episodes that end with the achievement of the goal. The percentage increase from $\alpha = 0.1$ to $\alpha = 0.2$ is immediately recognizable. We can also see that the after that the percentage of reward remain more or less equals. The increase in percentage from $\alpha = 0.1$ to $\alpha = 0.2$ is consistent with the mean rewards in figure 4.

Lastly we can also conclude that, despite the fact that increasing alpha we increase the rapidity of the learning, with Q-Learning is poorly influenced by changes in alpha in this environment. This can be inferred from the fact that the percentage of success is more or less equal for $0.2 \leq \alpha \leq 1$ and from the fact that the mean rewards remain more or less equals[4]

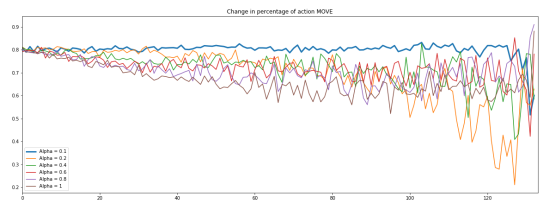
B. *SARSA (no softmax)*

In this test we try the SARSA algorithm with both an ϵ greedy decision policy. Since the poor results of this test the mean rewards were not reported.

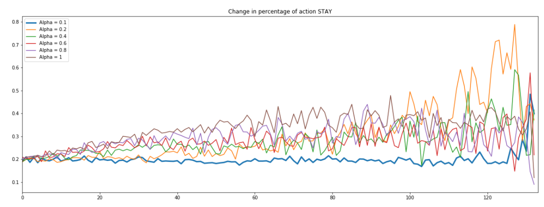
With this configuration the agent was unable to learn any type of complex behavior. This can be notices by the the percentage of movement and stay in figure 6. We can easily see that there is no convergence expect from the last episode

[3]In other case the influence of alpha is heavier and with more visible results.

[4]Also the increase in the mean reward from $\alpha = 0.1$ to $\alpha = 0.2$ is smaller than we can expect by looking the percentage of success.



(a) Percentage of action move (SARSA ϵ greedy)



(b) Percentage of action stay

Fig. 6: Percentage of actions stay and move (SARSA ϵ greedy)

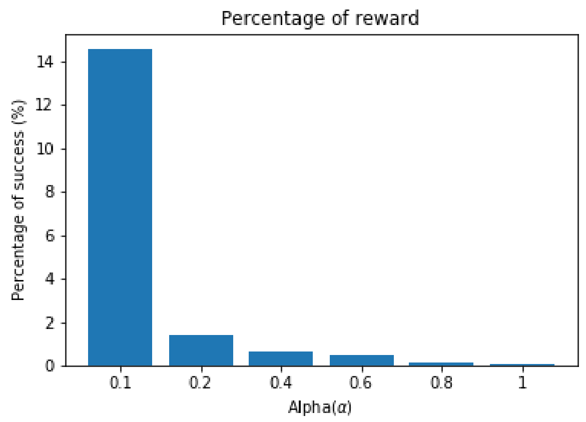


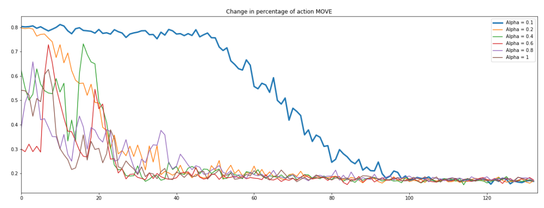
Fig. 7: Percentage of episodes with positive reward (SARSA ϵ greedy)

of the case $\alpha = 0.1$ (we can suppose that with more episodes the results will be better but in this case there will be no sense in the comparison with the other technique). The absence of any learning from the other alphas is also visible from the great variance of the percentage in last episodes, sign that the agent had not yet learn an optimal path.

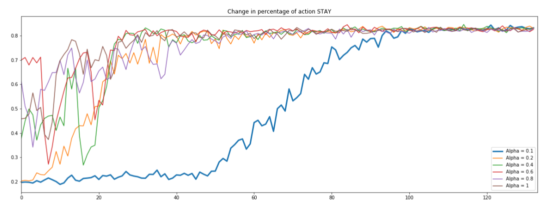
The percentage of success in figure 7 are consistent with the percentage of movement. Only $\alpha = 0.1$ seems to converge in last episodes and in fact the percentage of reward are much bigger in that case. An other important details to noticed in the percentage of rewards is the fact that despite $\alpha = 0.1$ had more probability of success respect the other cases the absolute value of its probability is very low (14%). This can be interpreted as another sign of no learning or very low learning.

C. SARSA (Softmax)

In this test we try the SARSA algorithm with both a softmax decision policy. This configuration was the one with the best results and the one used for future advance test.



(a) Percentage of action move



(b) Percentage of action stay

Fig. 8: Percentage of actions stay and move

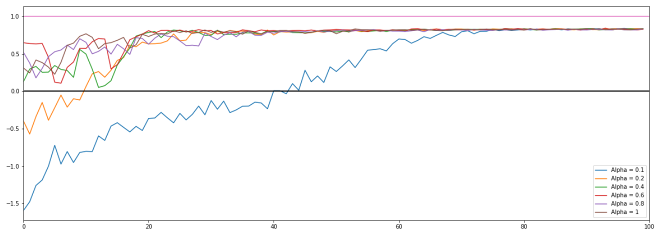


Fig. 9: Mean rewards (SARSA Softmax)

From figure 8 is immediately clear that in this case the increase of alpha has a lot of influence in the behavior of the agent. Also from the comparison of figure 8 with the other figure stay/move we could easily see that with this setting even the worst case ($\alpha = 0.1$) has an increase of the the stay action quicker respect the other settings.

The influence of alpha in the learning process can also be noticed by the trend of the various percentage. With $\alpha = 0.1$ and $\alpha = 0.2$ we can noticed a more regular trend towards the convergence. Instead with bigger alpha we note an high variance, especially in the initial episodes. This is consistent with the role of alpha in the the SARSA algorithm, since more alpha is close to 1 more important are the last information. Give more weight to the latest information means give more weight to exploration and in the extreme case learn only from exploration and not from the prior knowledge. In this case in some episodes the agent can be lucky and reach immediately the goal and in other wander for the environment all the time; this explain the high variability in the first episodes.

The analysis of the percentage of move/stay also suggest us that this agent learn very quickly and this fact is coherent with the mean reward in figure 9. From this figure we can easily seen that in general this configuration has a very high rewards and in all case converge towards the maximum rewards in few episodes (expect from $\alpha = 0.1$ where the convergence

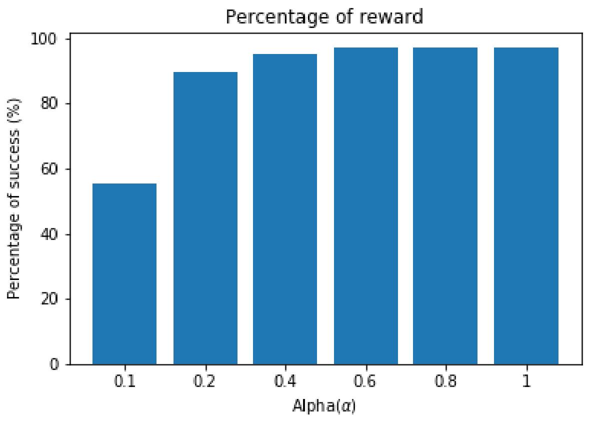


Fig. 10: Percentage of episodes with positive reward (SARSA Softmax)

is slower). The previous analysis about relation between the value of alpha and the trend of the curve is also valid with the mean reward. For $\alpha = 0.1$ and $\alpha = 0.2$ we increase in the mean reward is stable. Instead with higher alpha we have a high variance in the first episodes.

From figure 10 we can see that this configuration is far better that the other. Even the worst case has a success rate of almost 60%. With the other alpha the percentage increase rapidly and for $alpha \geq 0.6$ the percentage of success is very close to 100%.

D. Conclusion

From the test of various algorithm SARSA with a softmax decision policy is clearly the best results. With more epoch probably the other configuration could have been achieved better results but this will be another proof of the superiority of SARSA for this task.

In figure 11 can be seen the longest path for the various alpha which led to a reward. The goal is the green point and the blue line is the path. The direction is from the dark shades of blue to the light shades of blue. Unfortunately since the algorithm can go back on its steep in some cases the path can be a little bit confusing.

V. Results - Advance Implementation

In this section will be shown the results of evolution of the standard model. More precisely the test were focused the creation of an agent that will be able to reach multiple goal during a single episodes and the creation of an agent that will circumnavigate the obstacles instead of passing by.

A. Multi Goals Agent

The basic idea under this agent was to create an agent that have to reach intermediate goals before reaching the finale goal.

The fist implementation of multiple goal were unsatisfactory for two main reasons:

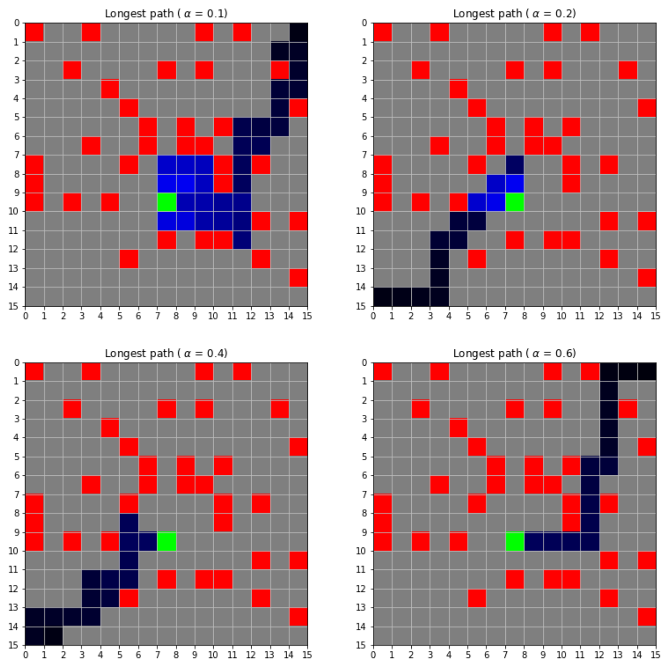


Fig. 11: Longest path during the training of the SARSA agent

- After reaching the first goal the agent was always rewarded to stay above the goal position. I.e. it had no reasons to move from the first goal.
- Q-table implementation.

The first problem was due to the implementation of the multiple goals and was resolved adding a penalty if the agent stay in a goal for more that a turn (expect from the final goal).

Unfortunately this lead to the second problem, the Q-table. Once the agent was forced to move from the goal reached it always derive its action from the old Q-table that contain the information to reach the old goal. So even taking into account the exploration the agent tends to go towards the old goal. This problem was resolved adding a new dimension to the Q-table. More precisely instead of having $(x*y) \times n.states$ we have $(x * y) \times n.states \times n.goal$. So every time a goal was reached the agent will pass to the subsequent level of the q-table. In this way we agent is capable of learn a path between the various goal[5].

Example of the results of this agent can be seen in figure 12.

B. Avoid Obstacles Agent

The basic idea under this agent was to create an agent that not only do not go over the obstacles but completely avoid, if is possible.

Respect the multi goals agent the implementation of the the avoid obstacle agent was much simpler and straightforward. The results were easily obtained with adding a penalty not only if the agent will hit an obstacles but even if the agent pass near an obstacles.

[5]For simplicity we maintain the same order of goal in every episodes. Furthermore change the order otherwise will have lead to led to the reappearance of the first problem.

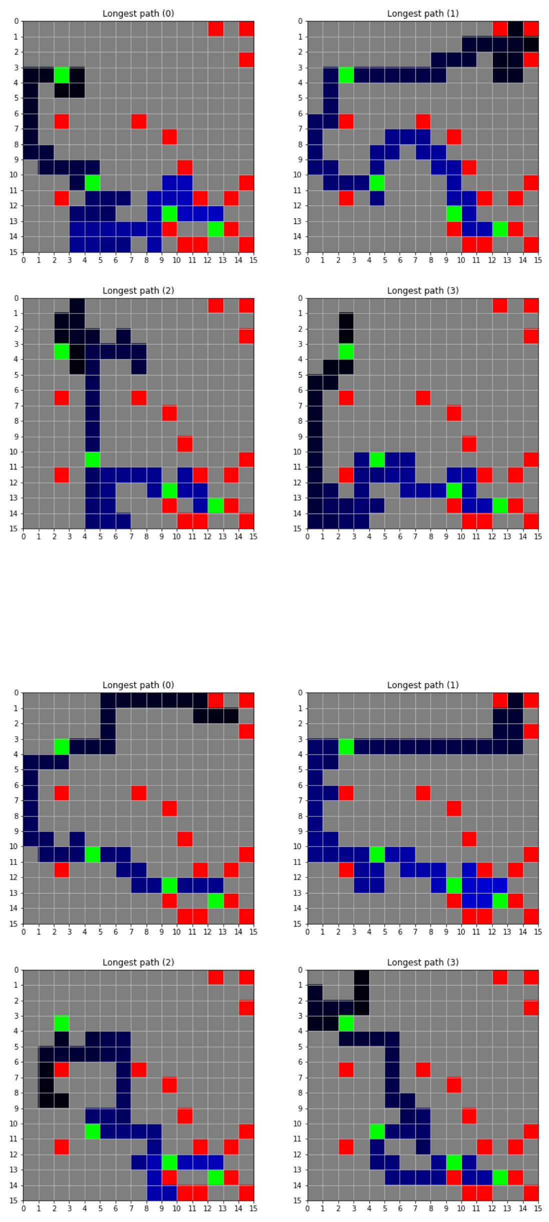


Fig. 12: Example of path with multiple goal

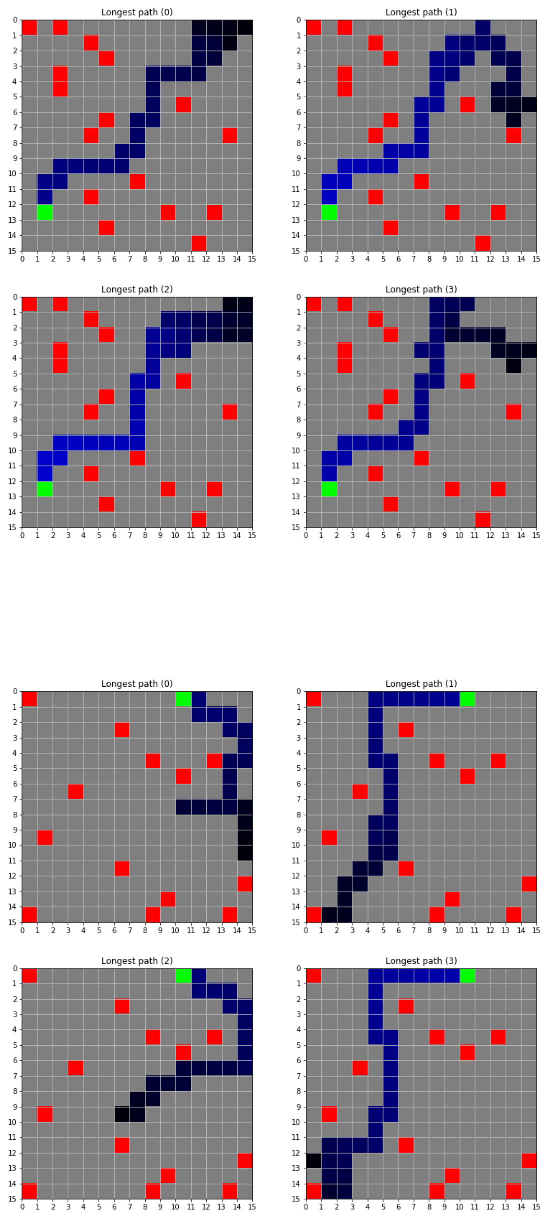


Fig. 13: Example of path with avoid agent

Since the obstacles layout is randomly generate, in the first test the agent often does not reach the goal. This because the unfortunate arrangement of obstacles can prevent the agent to reach some zone of the map since the agent cannot even stay for a turn near an obstacle. This was also a problem if the agent has the bad luck to spawn adjacent of an obstacles. This problem was resolve allowing the agent to stay near obstacles but at the same time giving him a penalty for such action. In this way passing near obstacle is not completely prohibited but strongly discouraged.

Example of the results of this agent can be seen in figure 13.