Modular Reinforcement Learning

Bharath Masetty

bmasetty@utexas.edu — bm37498

1 Introduction

Reinforcement Learning is a branch of machine learning used for applications where an agents needs to learn actions that maximize some form of cumulative reward. A Markov Decision Process (MDP) is a mathematical formulation used to represent a reinforcement learning problem. Temporal difference learning is a model-free class of reinforcement learning algorithms where the agent does not have access to the transition dynamics and performs bootstrapping using the current estimates of the state values. Q-learning is a popular temporal difference method used as a standard benchmark in many applications. In this assignment, we use a divide an conquer approach for solving a complex grid-world task by dividing a single task into multiple sub-tasks or modules then evaluate parallel Q-tables for each task. We then choose an action by using ϵ-greedy policy over a weight normalized Q-table.

The task requires the agent to learn to reach the terminal states in the 6 x 25 grid by 1) following a sidewalk, 2)avoiding the obstacles and 3) collecting the litter. A simple illustration of the grid-world is provided in Figure[1]. The green path in the center represents the side-walk and blue states on the right represents the terminal states. The states in red and orange represent obstacles and litter respectively.

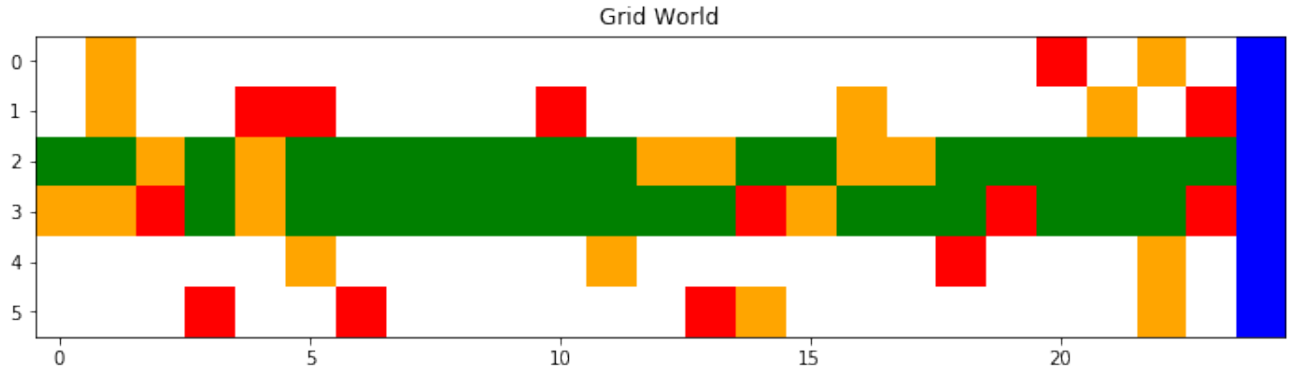


Figure 1: Sample layout of the grid-world environment.Orange(Litter),Red(Obstacles),Blue(terminal states),Green(side walk).

.

To solve this task, we divide the problem into 4 different modules and learn q-tables for the state-action space of each module. We demonstrate the behaviour of the agent in each individual module along with the combined behaviour in the following sections.

2 Methods

2.1 State and Action Space

We break up the problem into 4 different modules of reaching the goal faster(M1), staying on the sidewalk(M2), avoiding the obstacles(M3) and collecting litter(M4) respectively. The action space of the agent consists of 3 actions right, up and down represented by 0, 1, and 2 respectively. The state space of M1(goal) and M2(sidewalk) is equal to the number of states in the grid. For M3 and M4, we use a 3 digit binary string to represent the agent's state. The presence of '1' at an index in the string represents the presence of an obstacle or litter for that particular action. This way, the state space for M3 and M4 can be condensed to 8 unique states ranging from 0 to 2^3 in decimal form.

2.2 Reward Function

The reward function for M1(Goal) is defined as follows:

$$r_1(s, a, s') = \begin{cases} 10 & \text{if agent enters terminal states.} \\ -100, & \text{If agent moves into grid boundaries.} \\ -10, & \text{Otherwise} \end{cases}$$

The reward function for M2(side-walk) is defined as follows:

$$r_2(s, a, s') = \begin{cases} 1 & \text{Moving right in the side-walk..} \\ -10, & \text{Otherwise} \end{cases}$$

The reward function for M3(Obstacles) is defined as follows:

$$r_3(s, a, s') = \begin{cases} -10 & \text{Moving into the obstacle.} \\ 3, & \text{Moving towards the goal without obstacle.} \\ -2, & \text{Moving away from the goal without obstacle.} \end{cases}$$

The reward function for M4(Litter) is defined as follows:

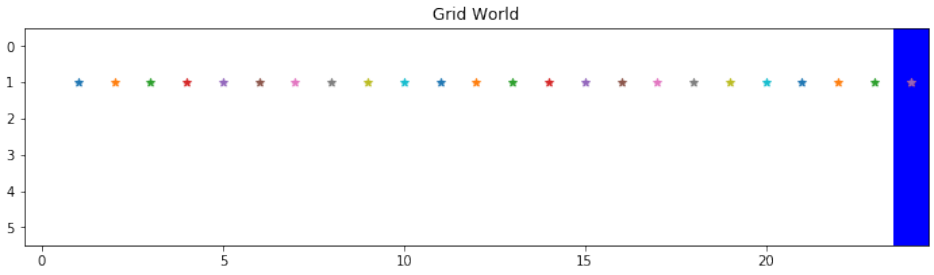
$$r_4(s, a, s') = \begin{cases} 5 & \text{Moving into the litter.} \\ 0, & \text{Otherwise.} \end{cases}$$

2.3 Discount Factor

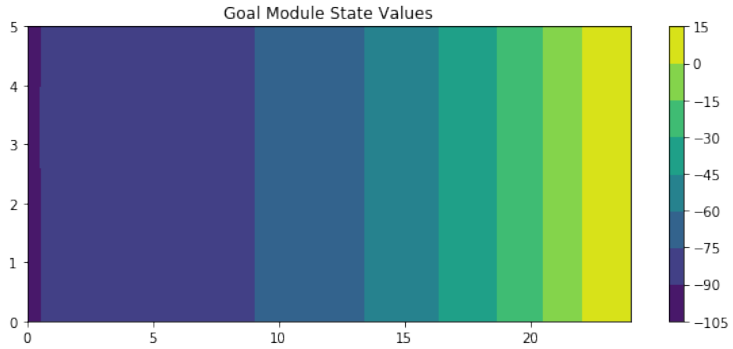
Every episode, the position of obstacles and litter is randomly initialized. For this reason, using a non-zero discount factor(γ) for M3 and M4 would result in undesirable state action values. For any given state, the resulting state upon taking an action changes randomly with uniform probability in every episode. Upon the setting the discount factor to 0 for M3 and M4, the agent learns to take the action with highest single step reward. Since there are other modules that can guide the agent to the terminal states, this configuration would result in q values that can tackle the stochasticity of the environment. A γ value of 0.9 is used for M1 and M2. The learning rate is set at 0.1 for all four modules. In all four cases, the action output of a single module is chosen based on an ϵ-greedy policy with an $\epsilon = 0.01$

3 Results

In this section, we present the resultant behaviour of the agent using each individual module and also a combination of all 4 modules. Figure[2] shows the results of the q - learning agent following the goal module only. As expected the, the agent learns to move to the right always and the values of the states increases as we reach the terminal states as shown in Figure 2(b).



(a)



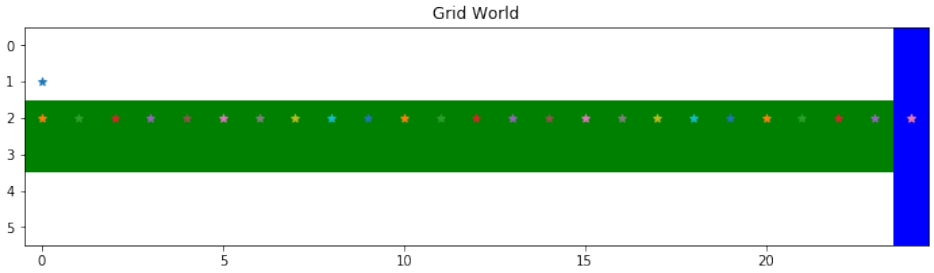
(b)

Figure 2: (a) Trajectory of the agent after '1000 episodes in M1(Goal) only. (b) State values for M1.

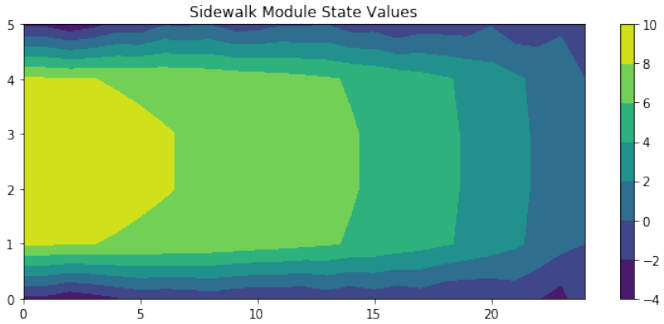
Figure 3 shows the behaviour of the agent following only the side-walk (M2) module. The agent eventually learns to move towards the goal while being on the sidewalk. The contour plot shown in Figure 3(b) shows that the values of the states towards the side-walk is relatively higher than the values of states away from the side-walk.

Figure 4 shows the behaviour of the agent following only M3(obstacles) module. The agent demonstrates the expected behaviour of avoiding the obstacles. The value of all 8 states in M3 is shown in Figure 4(b). It can been seen that the values of the states surrounded by obstacles (111 or 7.0) is the lowest and the states with no obstacles (000 or 0.0) is the highest. This shows that the policy for M3 is near optimal.

Similarly, the results in Figure 5 shows the behaviour of the agent in following only the litter (M4) module. It can be seen that the agent learns to collect as many litter objects as possible on its way to the terminal states. Figure 5(b) also shows that the values of states surrounded by litter (111 or 7.0) is much more then states without litter (000 or 0.0). This suggests that the agent's behaviour is near optimal in this module.

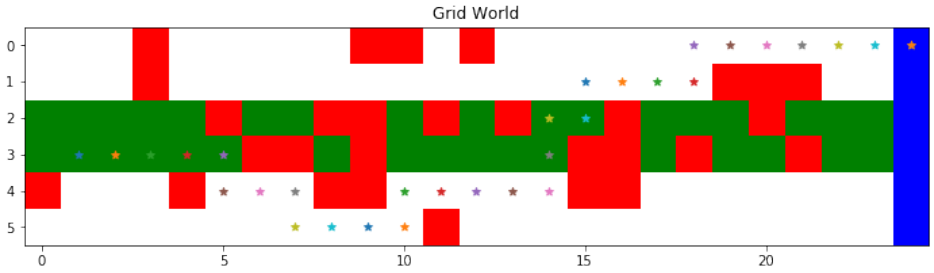


(a)

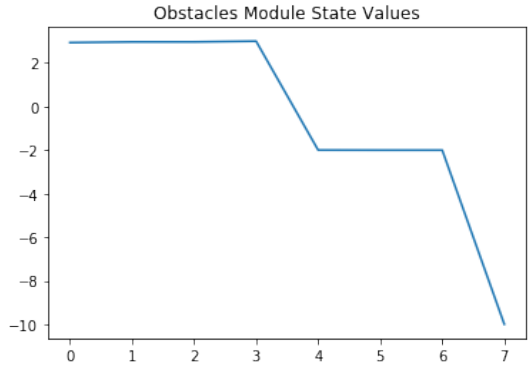


(b)

Figure 3: (a) Trajectory of the agent after ‘1000 episodes in M2(side-walk) only. (b) State values for M2.

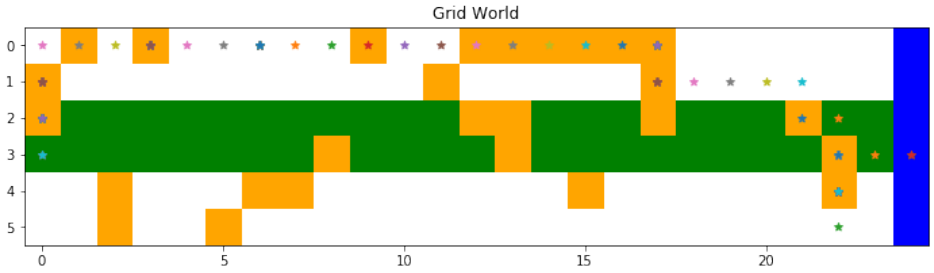


(a)

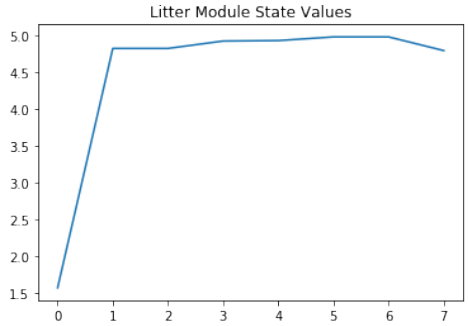


(b)

Figure 4: (a) Trajectory of the agent after ‘1000 episodes in M3(obstacles) only. (b) State values for M3.

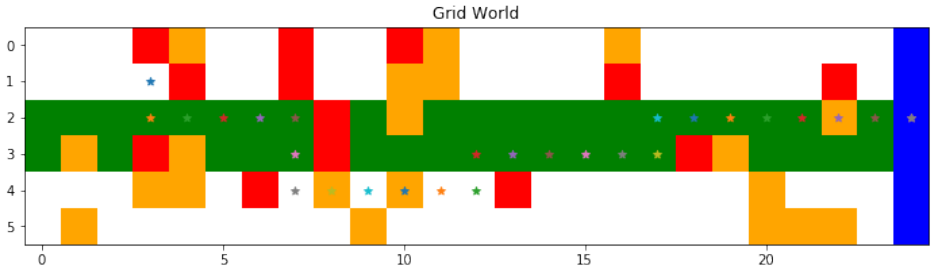


(a)

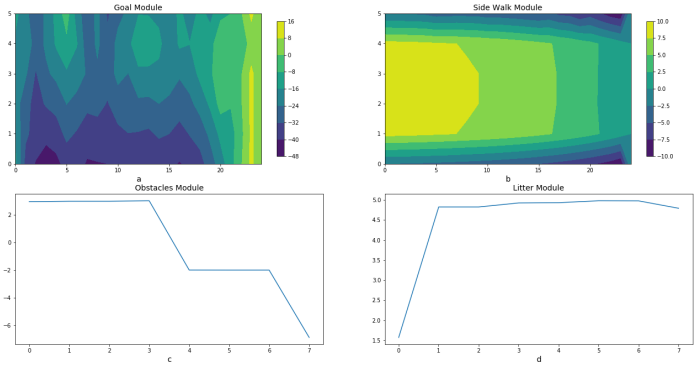


(b)

Figure 5: (a) Trajectory of the agent after ‘1000 episodes in M4(litter) only. (b) State values for M4.



(a)



(b)

Figure 6: (a) Trajectory of the agent using all 4 modules. (b) State values for all 4 modules.

Finally, we now move on to analyse the combined behaviour of all 4 modules. Figure 6(a) shows the trajectory of the agent after 1000 training episodes. It can be observed that the agent tries to avoid the obstacles and pick up the litter by progressing through the side-walk as much as possible. Figure 6(b) shows the state values of all four modules for this case. It can be observed that the state values are not perfect but close to the patterns observed in individual module cases. Therefore the desired behaviour has been achieved by combining all four modules.

4 Conclusion

In this assignment, we have implemented a simple version of modular reinforcement learning using q-learning to learn the state-action values of each individual module. The performance of the agent using individual modules satisfies the respective purposes. The final combination also demonstrates expected behaviour of reaching the goal through the sidewalk by picking litter and avoiding obstacles simultaneously.

The main advantage of this modular approach is an overall **reduction in dimensions of the state space**. Given that the environment is changing every episode, the size of the state space would be incredibly large to account for all possible states (6 X 25 X 8 X 8 = 9600 states). Since we used a modular approach, the total size of the state space is simply the sum of number of states in each individual module (6x25 + 6x25 + 8 + 8 = 166 states). It can be seen that the state space is reduced drastically. This offers a huge computational advantage since reinforcement learning suffers from the **curse of dimensionality**. Overall, we have demonstrated how a modular approach towards reinforcement learning can be used to efficiently learn a complex task using a reduced state space on a 2-dimensional grid-world environment.