Vignesh Ravindranath
vgr325

Reinforcement Learning

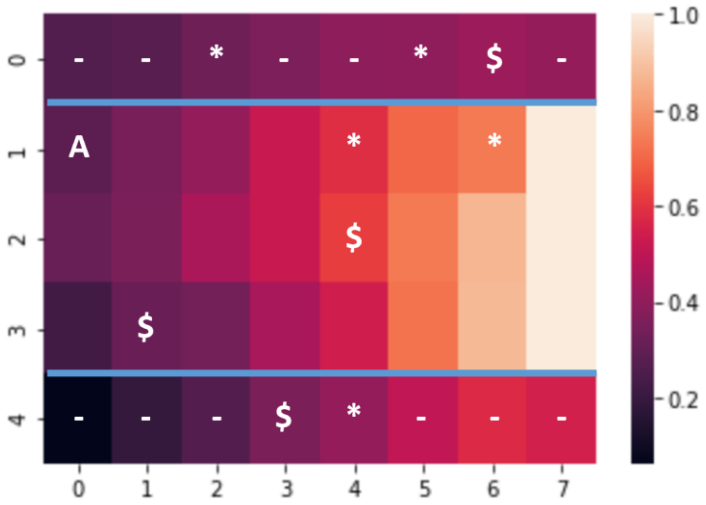
1 Overview

The purpose of this assignment is to implement a grid world game, where an agent must learn to walk down a sidewalk (2D grid) while avoiding obstacles (*) and picking up litter ($) using reinforcement learning. Reinforcement learning (RL) is a machine learning technique that uses a reward function and probabilistic dynamics to allow an agent to explore and learn its environment through various iterations. For this assignment, an agent will learn three different modules – sidewalk, obstacle, and litter – via Q-learning.

2 Algorithm and Methods

2.1 Q-Learning

In the grid world game, an agent (A) is placed on a 2D grid. Some squares contain rewards (positive or negative values) that are unknown to the agent at first. At each step, the agent can take an action (move up, down, left, right) to move from one state to another state. The agent must stay on the board at all times. As the agent moves across the board, it will learn where via a reward function, the best path to the end of the sidewalk. For a quick overview, the board is a 5×8 grid with an agent (**A**) starting at position (1,0). The sidewalk consisted of rows 1-3; rows 0 and 4 were not the sidewalk (denoted by -). Obstacles (*****) and litter (**$**) were placed randomly between columns 1 and 6. See image below for an example.



RL consists of a set of states, a set of actions, a reward function, and a state transition function. For this assignment, the set of states are the individual cells, and the set of actions are the

movements the agent can take at any given step. The state transition function is a probability the agent moves from on state to another via some action.

Q-learning is a model-free RL algorithm that can be used to train our agent to traverse this environment. The Q-table consists of values at each state for each possible action. The largest value corresponds to the action that will provide the maximum value of the next step. See image below [1].

To train a Q-table we use the Bellman equation (temporal difference learning) stated below [1].

$$Q^{new}(s_t, a_t) = Q(s_t, a_t) + \alpha \cdot \left(r_t + \gamma \cdot \max Q(s_{t+1}, a) - Q(s_t, a_t)\right)$$

The left-hand side is the new Q-value. The right-hand side consists of the old Q-value, α the learning rate, r_t the reward when moving from state s_t to s_{t+1}, and γ the discount factor.

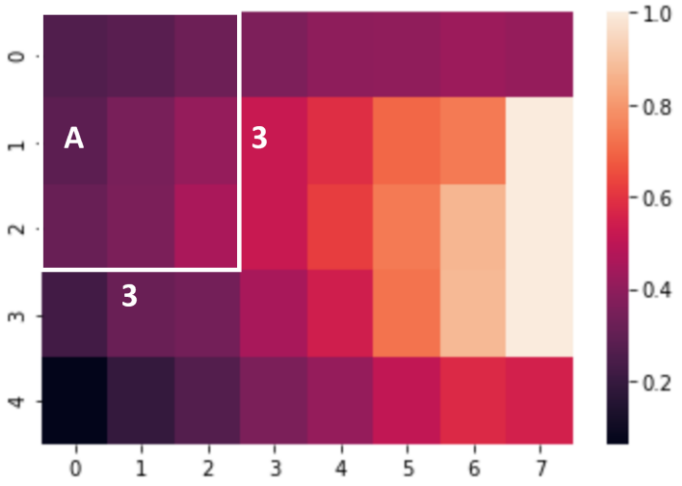
Rather than training the agent on one Q-table that incorporates all possible objects (non-sidewalk, obstacles, and litter), we can train the agent on individual modules and linearly combine the three together. For all modules, the action chosen was determined via a greedy method; the agent would use the corresponding Q-table to choose an action with the maximum expected value. Steps were non-deterministic, however, so there was a chance the agent would not take the 'best' action for a given state to encourage exploration.

2.2 Sidewalk Module

The sidewalk module was trained to reward the agent (+1) when reaching the end of the sidewalk – (1,7), (2,7), (3,7) – and reward the agent (-1) when walking off the sidewalk.

2.3 Obstacle Module

For the obstacle module, the agent had a 3×3 window to view its surroundings. If objects were within the window at a given state, the agent would be penalized based on a distance-based reward function between 0 and 1. If the agent landed on a cell with an obstacle on it, the agent would be severely penalized (-1, -10, or -100) and the agent would have to restart its training.



2.4 Litter Module

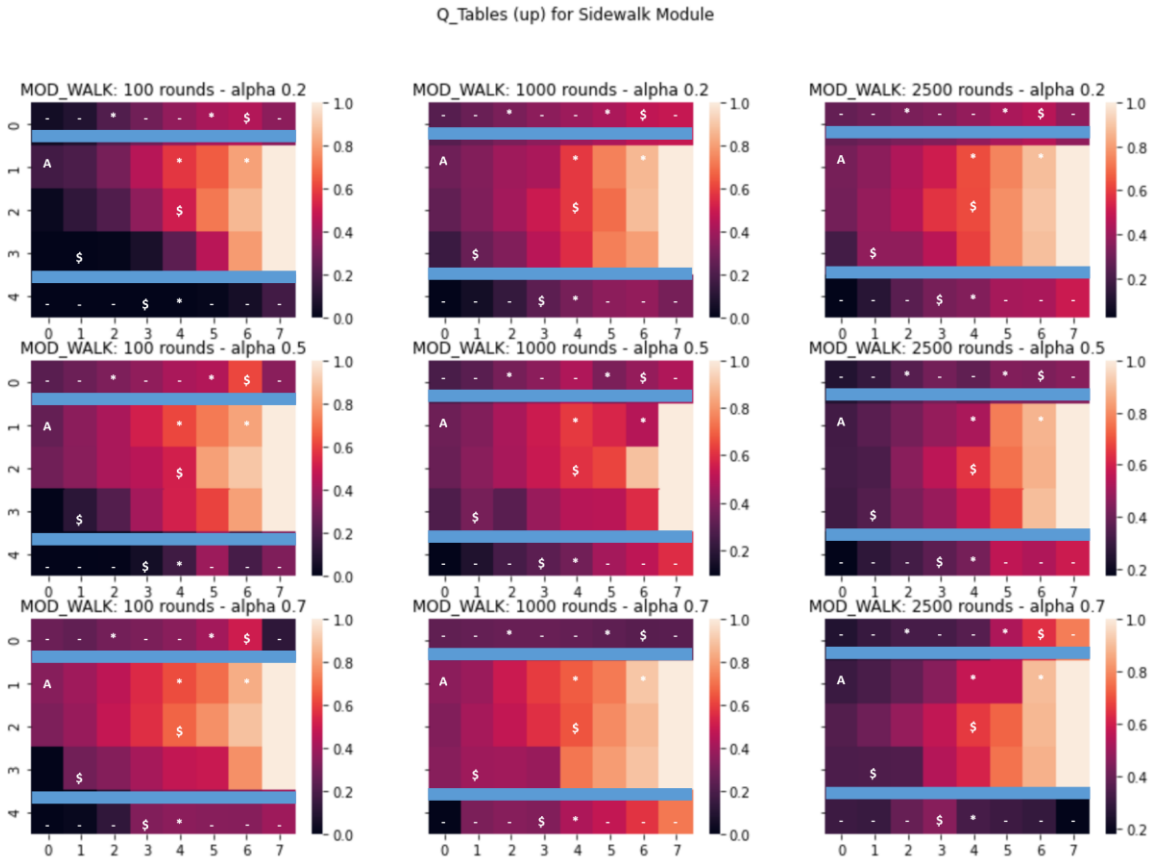
For the litter module, the agent again had a window to view its surroundings. If litter were within the window at a given state, the agent would be rewarded based on a distance-based reward function between 0 and 1. If the agent landed on a cell with litter on it, the agent would be rewarded (+1, +10, +100).

2.5 Libraries

a) numpy
b) matplotlib – pyplot
c) seaborn
d) random
e) scipy – spatial.distance

3 Results

The modules were trained with varying learning rates and rewards as seen below. The learning rate determines the extent at which information overrides old information. First the sidewalk module was trained on three learning rates after 100, 1000, and 2500 rounds of training.

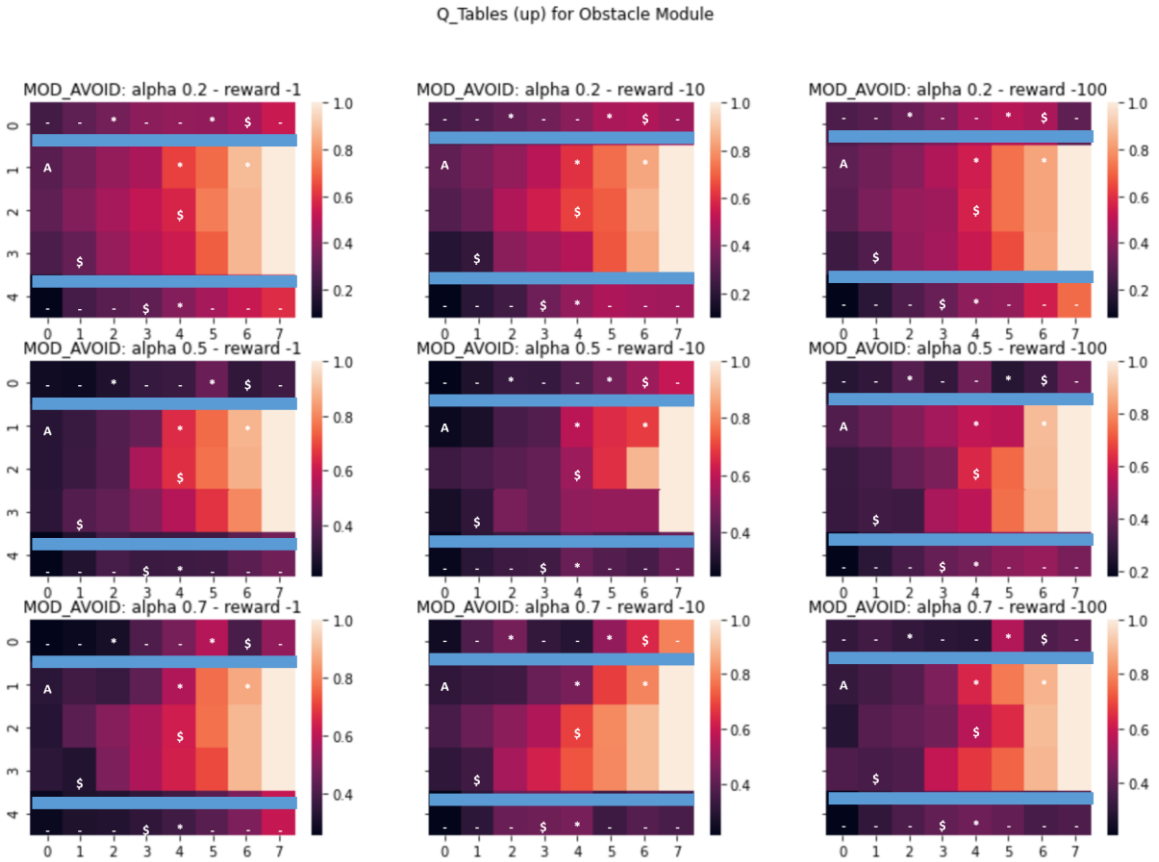


Above is a heat map of the Q-table for the sidewalk module for the `up` action. The darker regions indicate a lower Q-value while brighter regions indicate a higher Q-value. The heatmaps indicate

that the agent is learning to stay off of the sidewalk (Q-table is in the Jupyter notebook). The final reward is listed below.

	100 Rounds	1000 Round	2500 Rounds
Learning Rate = 0.2	0.0263	0.2842	0.3160
Learning Rate = 0.5	0.2586	0.3441	0.3158
Learning Rate = 0.7	0.3373	0.2330	0.3051

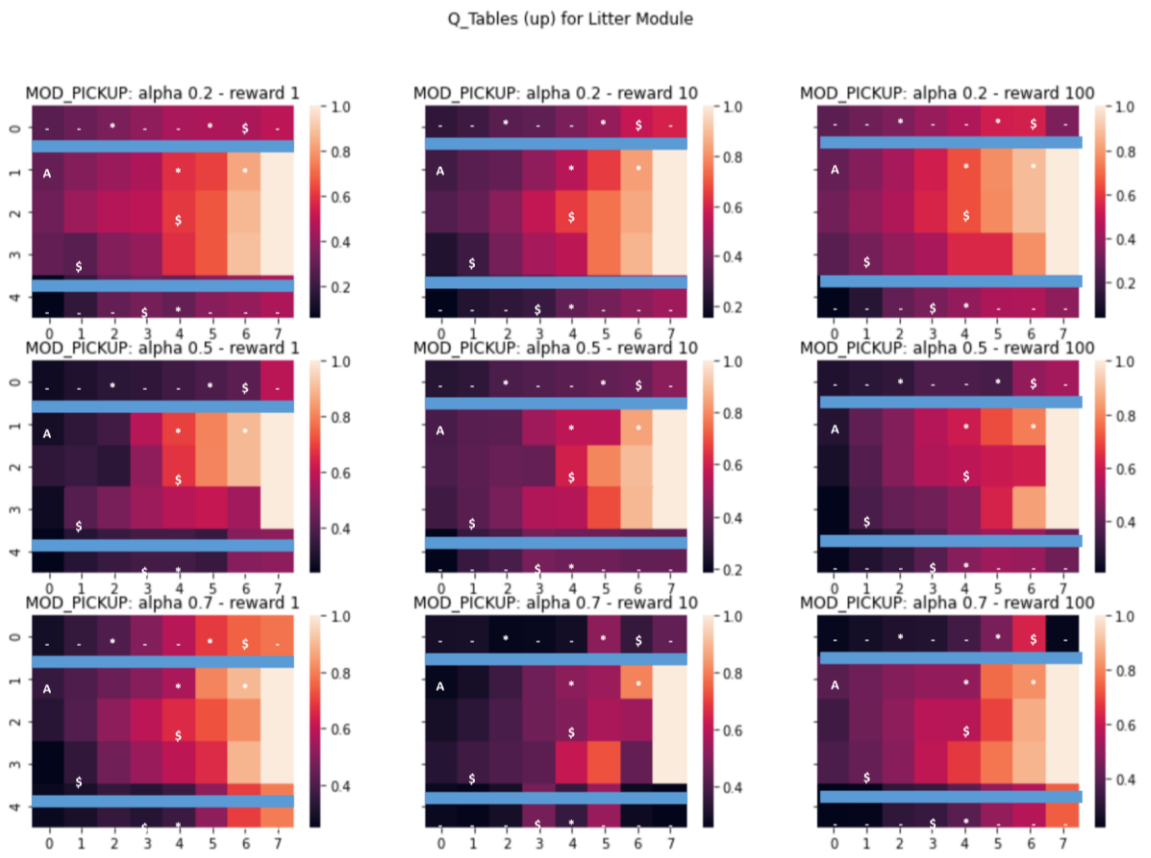
Next, the obstacle module was trained on varying alpha values and rewards after 2500 rounds of training.



Above is a heat map of the Q-table for the obstacle module for the `up` action. The heatmaps indicate that the agent is learning to avoid obstacles (Q-table is in the Jupyter notebook). The final reward is listed below.

	Reward -1	Reward -10	Reward -100
Learning Rate = 0.2	0.2609	0.2611	0.3028
Learning Rate = 0.5	0.2970	0.2647	0.3496
Learning Rate = 0.7	0.3416	0.3123	0.3199

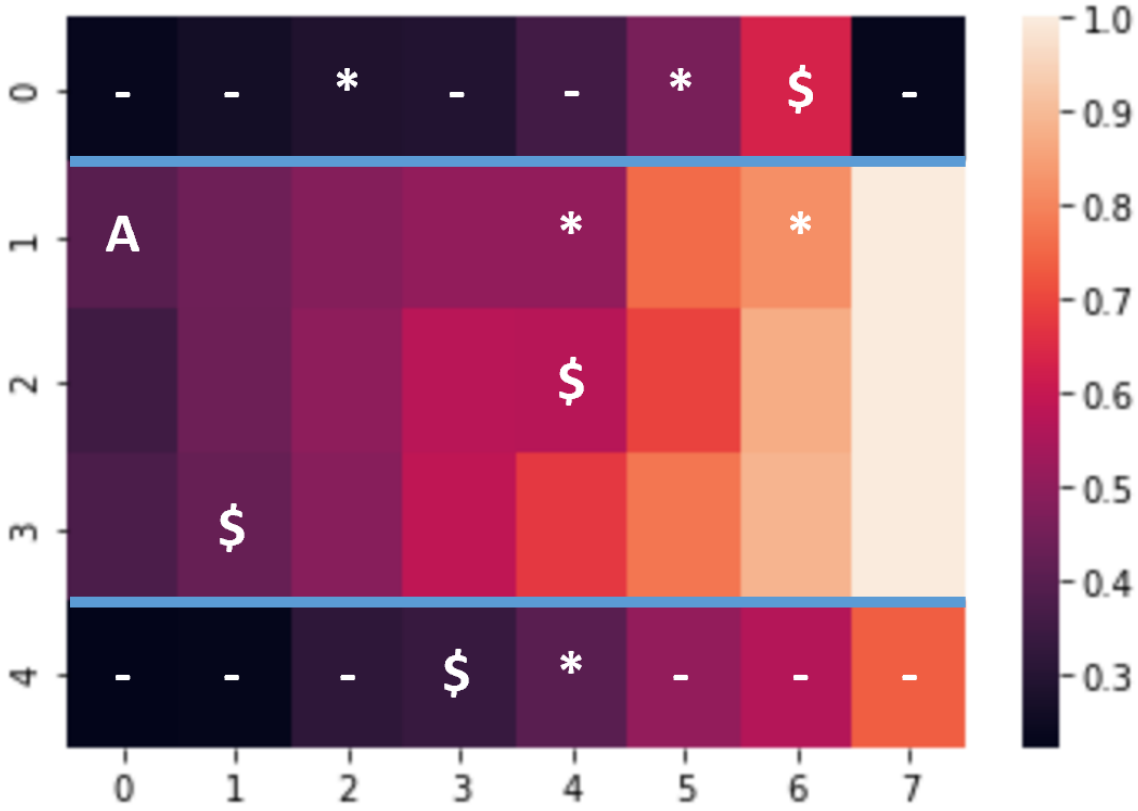
Lastly, the litter module was trained on varying alpha values and rewards after 2500 rounds of training.



Above is a heat map of the Q-table for the litter module for the `up` action. The heatmaps indicate that the agent is learning to go to litter (Q-table is in the Jupyter notebook).

	Reward +1	Reward +10	Reward +100
Learning Rate = 0.2	0.3058	0.2940	0.2923
Learning Rate = 0.5	0.2800	0.3394	0.2873
Learning Rate = 0.7	0.3590	0.2666	0.3999

Below is a heatmap of a linear combination of the three Q-tables for the `up` action.



4 Summary

Overall, Q-learning is an effective algorithm for a model-free RL problem. It works well when the agent has a finite number of actions to take in an environment. Moreover, it is effective at dividing the problem into separate modules that can be fine-tuned and linearly combined together.

5 References

[1] https://en.wikipedia.org/wiki/Q-learning

[2] https://towardsdatascience.com/implement-grid-world-with-q-learning-51151747b455

[3] http://karpathy.github.io/2016/05/31/rl/