Grid World Agent Simulation using Reinforcement Learning

Daniel Alejandro Noble Hernandez
dan833
The University of Texas at Austin

Introduction

The grid world game is a popular introductory problem in reinforcement learning, consisting of a 2-dimensional grid in which an agent resides. The agent begins in a certain state of the space, and it then behaves in a way defined by the program. The agent's actions consist of exploring and exploiting the grid to maximize the rewards given both by immediate actions and in the long run.

The purpose of this project was to implement the grid world game using the q-learning policy learning algorithm and the epsilon-greedy algorithm to determine the actions that the agent takes throughout the training process. At a high level, the q-learning algorithm utilizes a matrix called a q-table storing the values of the corresponding actions with those of each possible state; within that, the program utilizes the epsilon-greedy algorithm, choosing to explore the map with a probability less than some number epsilon, and choosing to exploit the rest of the time. Exploration instructs the agent to choose a random action and update the value of that action, whereas exploitation instructs the agent to choose the immediate action with the highest reward. The project yields maps of the agent's motion under four different models, that can be combined to find an overall suboptimal policy by which the agent should act.

Methods

The grid world was set as a 6 x 25 array, where each element represented a possible state that the agent could be in. Within that, the agent can take one of four actions: left, up, right or down. After this, the reward matrices, one for each of the different modules, were implemented, and they will be discussed in further detail within the subsections. Common to all subsections were the q-learning and epsilon-greedy algorithms, for which the steps were carried out as follows:

1. Set up the q-table with dimensions defined by the number of states and number of actions (150 x 4)
2. Store the current position in a path array and generate a random number between 0 and 1.
3. If the number found is less than epsilon = 0.1, choose a random action, checking for validity by ensuring that that action does not remove the agent from the grid. Otherwise, choose the action with the highest reward (as defined in the reward matrix), and similarly, ensure its validity.

4. Update the agents position, which is represented in code both in one dimension (a 1 x 150 array) and as a pair of coordinates for convenience.
5. Update the q-table as $Q(s,a) = (1 - \alpha) * Q(s, a) + \alpha * (R(s, a) + \gamma * \max(Q(s',:)))$ where s is the current state, a is the chosen actions, s’ is the next state, α is the learning rate (0.01), and gamma is the discount factor (0.6)
6. Loop through steps 2-5 until the agent reaches the right column of the grid.

1. Forth Module

The forth module is defined by a state space in which there are 150 grids and 4 actions that can be taken for each of these. It utilizes a reward matrix defined as follows:

- Moving left: R(s, a) = -5
- Moving right: R(s, a) = +5
- Moving into an end state: R(s, a) = +100
- All other motions: R(s, a) = 0

2. Sidewalk Module

The sidewalk module utilizes the same state space as the forth module, and its uses the following reward matrix:

- Moving from the sidewalk to its border: R(s, a) = -10
- Moving along the sidewalk border: R(s, a) = -10
- Moving from sidewalk border onto sidewalk: R(s, a) = +5
- Moving right: R(s, a) = +5
- Moving left: R(s, a) = -5
- Moving into an end state: R(s, a) = +100
- All other motions: R(s, a) = 0

3. Obstacles Module

The obstacles module also defined the state space as described above, but used the following reward matrix:

- Moving into an obstacle: R(s, a) = -10
- Moving right: R(s, a) = +5
- Moving left: R(s, a) = -5
- Moving into an end state: R(s, a) = +100
- All other motions: R(s, a) = 0

Note that moving into an obstacle did not remove the obstacle.

4. Litter Module

The litter module had the same state space as the previous modules, but used the following reward matrix:

- Moving into a litter space: R(s, a) = +10

- Moving right: R(s, a) = +5
- Moving left: R(s, a) = -5
- Moving into an end state: R(s, a) = +100
- All other motions: R(s, a) = 0

As in the obstacle module, stepping into a litter space did not remove the litter.

Results

1. Forth Module

The results from the forth module are shown in Figure 1. In this module, the agent starts on the left side of the grid with the aim of moving to the right side; the path taken is shown in Figure 1, with one kink attributable to the agent choosing to explore rather than exploit.

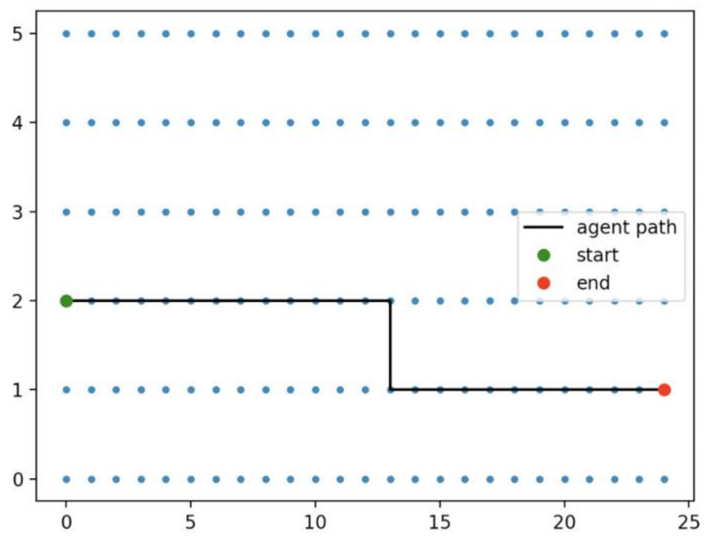

Figure 1: Forth module: Agent starts on the left side of the grid, with a goal of reaching the right side of the grid

2. Sidewalk Module

Figure 2 shows the results from the sidewalk module, in which the agent begins on the left side of the grid with the aim of making it to the right side of the grid without stepping off of the sidewalk. In this case, the agent stays within the sidewalk the whole time; this is mostly attributable to luck, as one would expect the agent to exploit at least once, in which case it would be likely that the agent would step onto the border of the sidewalk. Of course, it is possible that the agent exploited but remained on the path by chance.

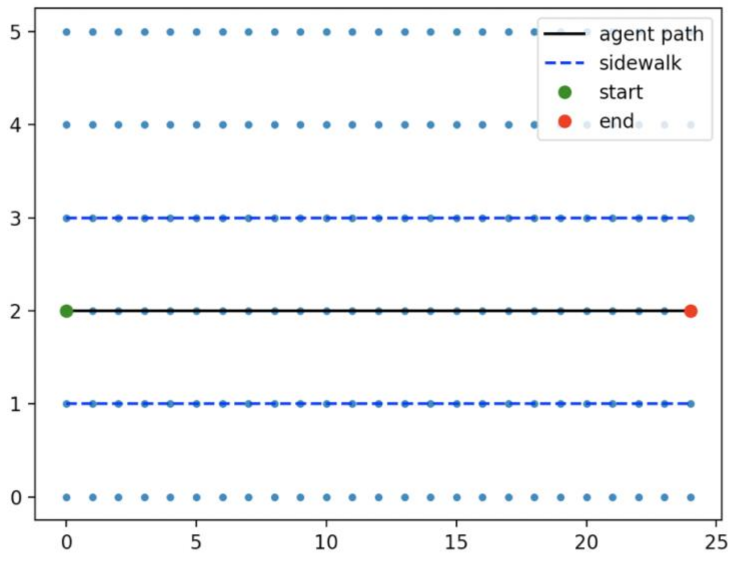

Figure 2: Sidewalk module: Agent starts on the left side of the grid, with a goal of reaching the right side of the grid without stepping off the sidewalk

3. Obstacles Module

Shown in Figure 3 below are the results of the obstacles module. This one exhibits significantly more turns in the agents motion – likely a result of the motivation to avoid obstacles, though also subject to the randomness of exploration – though it is driven primarily by an overall push towards getting to the right side of the grid.

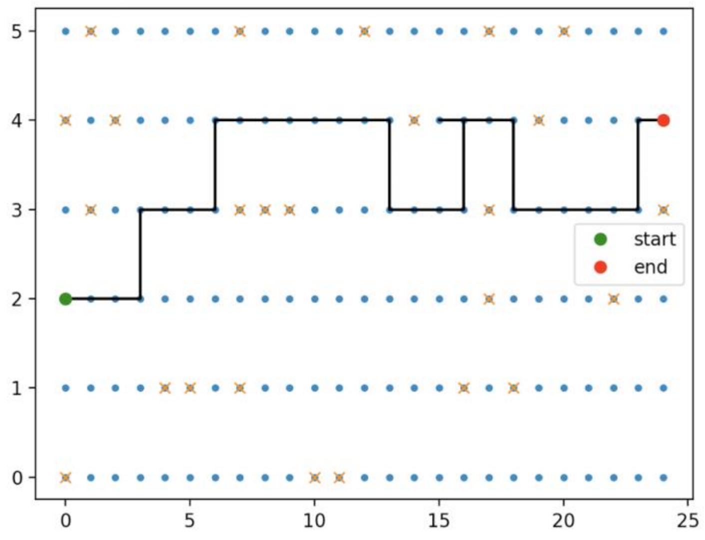

Figure 3: Obstacle module: Agent starts on the left side of the grid, with a goal of reaching the right side of the grid whilst avoiding obstacles (marked as x's)

4. Litter Module

The litter module results are shown in Figure 4, which looks similar to Figure 3; however, a crucial difference now is that though agent is overall driven by a rightward motion to the opposite side of the grid, it now makes an effort to pick up litter. This goes so far as to divulge from the most straightforward path, taking a roundabout trail in the effort to land on as many litter squares as possible as defined by both the algorithm and the reward matrix. Once more, slight deviations from this expectation are attributable to the exploration aspect of the algorithm.

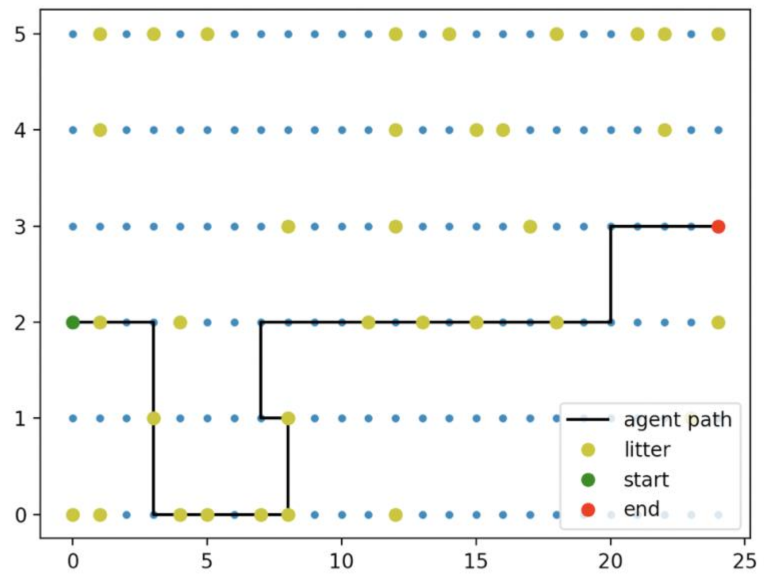

Figure 4: Litter module: Agent starts on the left side of the grid, with a goal of reaching the right side of the grid whilst picking up litter

---

Conclusion

Overall, the agent behaved as expected for each of the randomly generated maps of the forth, sidewalk, obstacle, and litter module; this led to an optimal policy being generated for each of these cases, with the ultimate idea that a suboptimal policy could be drawn from a combination of these optimal policies, perhaps using the softmax function. This would give a general best policy for a grid world consisting of each of these aspects. Unfortunately, the author had trouble combining the four policies and elected to leave that final aspect out of the code and report, but given more time, the foundational blocks for this procedure were in place and perhaps could have been done effectively. Nonetheless, as far as each of these individual modules go, the Q-learning and epsilon-greedy algorithms were effectively implemented to come up with a best policy for each individual case, and the results reflect what one might expect to see in each of those cases.]