Q-Learning, A* report

Alexander Baekey, April 17th 2021

https://github.com/alexbaekey/QLearningAstar

1 Game Description and Environment - Grid World

The objective of the game is for the agent to reach a goal using the two algorithms Q-Learning and A*. An occupancy grid is established to depict where the agent, end goal, and obstacles are. For the Q-Learning algorithm, these positions are updated for each timestep. For the A* algorithm, the grid simply provides position information for the entities on the grid. An example instance of the occupancy grid:

$$\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

where -1 indicates the position of the agent, -2 indicates the position of the goal, 1 indicates the position of an obstacle and 0 indicates an unoccupied tile. The goal is set to always be along the right side of the grid, the agent's starting position is randomized (excluding the far right column) and obstacles are randomized (excluding the positions of the goal and the agent). The starting position of the agent is reset for each episode. The agent is only allowed to move up, down, left, right, or to stay still.

2 Q-Learning

2.1 Time Steps and Movement

For the Q-Learning routine, the objective of the agent is to reach the goal tile within the least number of time steps. There is a maximum number of time steps set. If the agent does not reach the goal within the maximum number of time steps, the round is over and the agent is reset. There is also a set number of rounds called "episodes". Evaluation metrics are recorded per episode. For the Q-Learning algorithm, each movement in gridworld is stochastic. Each time an action is chosen there is a 60% chance of moving in the direction the action dictates. There is a 10% for each of the other 4 action possibilities.

2.2 Q-Learning methods

The Q value is calculated according to the equation:

$$Q^*(s,a) = Q(s,a) + \alpha(r + \gamma max_{a'}(Q(s',a')) - Q(s,a))$$

For the sake of establishing the Q-Learning table (Q-table), each element on the grid is treated as a "state" and each movement as an "action". The table is a 2D array with the dimensions (states,actions) since in my case I had 5 possible actions (up, down, left, right, or no movement) and 100 states representing the 100 tiles on the occupancy grid.

The three tune-able parameters of the Q-Learning algorithm influence how the Q-table is updated and the algorithm's ability to learn. Equation I contains two of these tune-able parameters. Alpha represents the learning rate and gamma represents the discount rate, which determines the importance placed on long-term rewards over short-term rewards.

The third parameter influences the agent's exploration rate. The Epsilon-Greedy algorithm used to decide which action is selected for each timestep. Epsilon dictates the probability of selecting a random action; otherwise the action with the maximum Q value is chosen.

2.3 Results

The values used for the Q-Learning session are as follows: number of obstacles=7, max time step per episode=200, number of episodes=100, max x=10, max y=10, alpha=0.9, gamma=0.9 With alpha set very low, the agent's performance was inconsistent, with little clear pattern towards reducing steps required to win. With gamma set very low, the agent reached the goal much less frequently overall and exhibited similar erratic behavior.

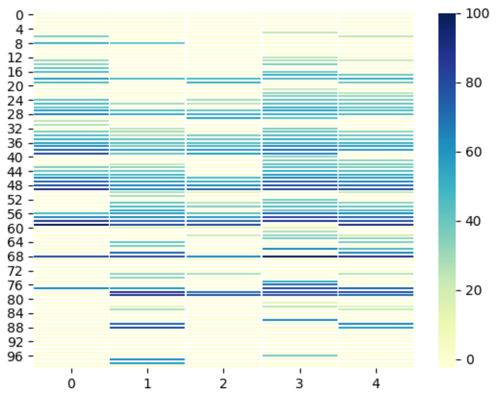
Figure 1: Heatmap of Q-table

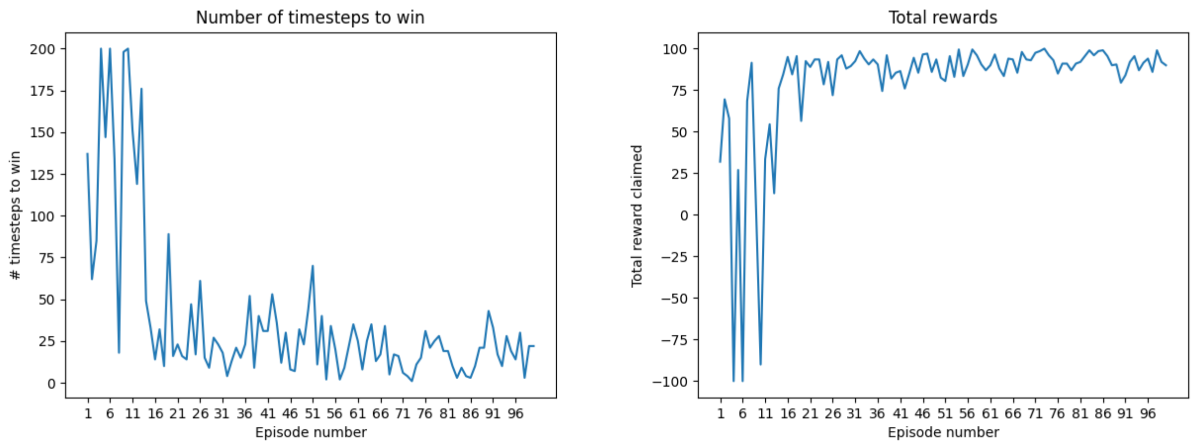

Figure 2: Q-Learning Evaluation

3 A*

The A* algorithm is a path-finding algorithm that uses a distance metric calculated for each position to step through a maze from a starting position to a goal position efficiently. The algorithm calculates adjacent values g, h, and f, then finds the minimum adjacent f value to select the next direction.

3.1 A* Methods

The g value is the Euclidean distance between the current position and the starting position. The same applies for the h value, except it is calculated from the current position to the goal. f is g+h.
Pseudo-code:

```
-Create arrays dim(max_x,max_y) for grid, g, h, f, and "visited" to avoid repetition. Initialize as 0, place obstacles.
-Create empty list for path from start to goal. Append current position.
-Run loop iterating over all adjacent spaces to current position
......-Check if reached goal, if so, break and append goal to list
......-If within bounds and not an obstacle, calculate g, h and f for current position. Save to arrays
-Call move function to choose direction to move by finding minimum adjacent f value
-Artificially reduce f value surrounding goal to make up for directional movement problem (no diagonal movement means agent strays away when it reaches the goal from a diagonal position)
-Update current position when goal is reached, append to path
```

3.2 Results

The results from a run of this implementation of the A* algorithm are as follows:
note: y values are flipped when printed, the top row is y=1, bottom is y=10
As before, -1 indicates agent, -2 indicates goal, 1 indicates obstacle
START: (6,10) END: (10,4)
PATH: [(6, 10), (6, 9), (7, 9), (7, 8), (8, 8), (8, 7), (8, 6), (8, 5), (9, 5), (10, 5), (10, 4)]

$$grid: \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$g: \begin{bmatrix} 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 6.08 & 6.32 & 6.71 & 7.21 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 5.10 & 5.39 & 5.83 & 6.40 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 4.12 & 4.47 & 5.00 & 5.66 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 3.00 & 3.16 & 3.61 & 4.24 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 2.24 & 2.00 & 2.24 & 2.83 & 3.61 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 1.41 & 1.00 & 1.41 & 2.24 & 3.16 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 1.00 & 2.00 & 0.00 & 0.00 \end{bmatrix}$$

$$h: \begin{bmatrix} 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 3.00 & 2.00 & 1.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 3.16 & 2.24 & 1.41 & 1.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 3.61 & 2.83 & 2.24 & 2.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 5.00 & 4.24 & 3.61 & 3.16 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 6.40 & 5.66 & 5.00 & 4.47 & 4.12 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 7.07 & 6.40 & 5.83 & 5.39 & 5.10 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 7.81 & 7.21 & 6.71 & 6.32 & 0.00 & 0.00 \end{bmatrix}$$

$$f: \begin{bmatrix} 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 9.08 & 8.32 & 4.71 & 4.21 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 8.26 & 7.62 & 4.25 & 4.40 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 7.73 & 7.30 & 7.24 & 7.66 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 8.00 & 7.40 & 7.21 & 7.40 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 8.64 & 7.66 & 7.24 & 7.30 & 7.73 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 8.49 & 7.40 & 7.25 & 7.62 & 8.26 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 8.81 & 7.21 & 7.71 & 8.32 & 0.00 & 0.00 \end{bmatrix}$$