Ta, Alton

ME 493

March 7th, 2017

Project Beta Report

**Program Overview**

The program uses two classes: an agent and a domain. The agent contains the learner, whose key properties are its alpha, epsilon, gamma, and starting locations. The domain contains the state representation, the Q-table, and the reward vector. The state grid can be customized with X columns by Y rows. The agent may move from cell to cell. If the agent tries to move off the grid, the agent’s movement will be cancelled.

The reward is based on which state the agent ends in. If the agent is able to change to a state that is not the goal, it is rewarded with -0.99. If the agent does not change its state (it hits a bumper), the agent is reward with -1. If the agent is able to change to the goal’s state, it is rewarded with 100.

The autonomous portion of the agent can be characterized with four functions: decide, act random, act greedy, and react. Decide determines if what the agent does is greedy or random by comparing ATRAND to the difference of 1 – epsilon. If ATRAND is greater than the difference, the agent will move in a random direction. Otherwise, the agent will find the maximum Q-value for the state it is currently in to make a greedy movement. After either a random or greedy action is taken, the agent reacts. It uses the previous state and action’s Q-value, the reward it received from moving into its new state, the maximum Q-value possible in its new state, alpha, and gamma to determine what the new Q-value for its action it performed is. The step counter is increased.

The four functions listed above are placed in a while loop. The while loop ends when the agent’s coordinates match the goal’s coordinates. Once this happens, the step counter value is recorded in a vector, then reset. The agent’s coordinates are placed back at (0,0) while retaining an updated Q-table. The agent’s epsilon value is decayed to encourage more greedy actions for future episodes. A FOR loop and IF statement are used to see if any of the Q-values exceeded the goal’s reward. If the statement is true, then one episode is considered concluded. Overall, the program runs 1000 episodes. One statistical run is considered as 1000 episodes of data.

**Test D (Lines 630 to 637)**

Test D was carried out using a FOR loop and an IF statement. It checked every value of the final Q-table to see if any elements were greater than the goal reward. Occasionally, there would be some Q-values that were 0.001 greater than the goal reward. This was deemed acceptable, however. Otherwise, no Q-value is shown to be greater than the goal reward. As such, this test is deemed acceptable.

**Test E (Lines 621 to 627)**

Test E was carried out in int main(), however it could have been made a function. The program ends when a filler variable, check, is greater than 100. By default, check is set to 10. Once the agent’s coordinates match the goal’s coordinates, check is set to 200 to end the while loop. The step counter is pushed back, the step count is reset, and the agent’s coordinates are set to (0,0). The Q-table is not changed in anyway; the new agent will have access to the previous agent’s final Q-table. This test is deemed complete.

**Test F (Lines 624 to 626)**

Test F was conducted in Excel after averaging 30 statistical runs. For the trials, an optimal amount of steps to reach the goal was determined to be 18 in a 10x10 grid, where the agent starts at one diagonal corner and the goal is at the other diagonal corner. An IF statement is used to see if the agent’s steps are below the optimal number. If this is true, that means the agent skipped a cell or teleported. If this is true, the program crashes. Both an average and a median were performed to see how many steps the agent took. A median was taken to compare against the average. The median was expected to be lower, as the first few trials were exponentially greater than later trials, which would throw off the average. Two averages and medians were taken. The first includes the entire data. The second only includes the last 95% of the episodes. A pass/fail determination was chosen as: If the agent took two or more steps than necessary (11% greater) than the optimal amount of steps, the agent is not within optimal range. Using the testing constants shown in Table 2, the average and median steps to reach the goal are as follows.

Table 1: Different averages and medians for the conducted trial.

|  |  |
| --- | --- |
| Test type | Steps taken |
| Average, Full | 31.4 |
| Median, Full | 26.4 |
| Average, 95% | 26.3 |
| Median, 95% | 26.2 |

These are not optimal. However, they can be made optimal by changing the epsilon decay constant, as will be explained in the **Test Parameter Variation** section. With that in mind, this test will be considered a success with appropriate parameters. The program is successful in that the agent does not skip cells or teleport to the goal. The amount of steps take is always greater than or equal to 18 when using a 10x10 grid.

**Test G**

Test G was not performed.

**Learning Curve**

The learning curve was conducted such that one statistical run was 1000 episodes. 30 runs were conducted to negate random chance. The parameters shown in Table 2 show what values were used to conduct the 30 run statistical trial.

Table 2: Constants used for testing.

|  |  |
| --- | --- |
| Variable | Value |
| Alpha | 0.1 |
| Epsilon | 0.3 |
| Gamma | 0.9 |
| Epsilon decay constant (EDC) | -0.000001 |
| x (number of columns) | 10 |
| y (number of rows) | 10 |
| States | 100 |
| Ax (agent x starting position) | 0 |
| Ay (agent y starting position) | 0 |
| Gx (goal x position) | 9 |
| Gy (goal y position) | 9 |
| Non-goal Reward | -0.99 |
| Bumper Reward | -1 |
| Goal Reward | 100 |

Using the parameters in Table 2, 30 sets of data were generated. Each episode was averaged for all 30 runs. The learning curve for this trial can be seen in Fig. 1 below.

Figure 1: Agent learning curve for GridWorld.

Overall, it can be seen that the agent does take a relatively larger amount of steps for its first couple of dozen trials, peaking at around 1400 steps. At about the 50th episode, the agent’s number of steps has essentially converged. As noted in the **Test F** section, the average for all episodes was about 31.4 and the median was about 26.4. This learning curve looks appropriate for this situation. It is possible that the agent is learning too quickly, as the amount of steps taken dramatically decreases quickly.

**Test Parameter Variation**

The epsilon decay constant (EDC) was chosen as the varying test parameter. The equation to attenuate epsilon is as follows,

where i is the current episode. As stated in Table 1 earlier, EDC was set to -0.00001. It is negative to ensure decay. From how this equation works, it is speculated that as EDC becomes more negative, the amount of steps it takes will decrease. If it becomes more positive, the amount of steps it takes will increase. One higher and one lower EDC value will be tested. All other variables were kept the same as Table 2 to ensure that there are controlled variables.

EDC = -0.0001

Figure 3: Learning curve for a higher EDC.

Table 3: Averages and medians for a higher EDC.

|  |  |
| --- | --- |
| Test type | Steps taken |
| Average, Full | 25.1 |
| Average, 95% | 19.2 |
| Median, Full | 18.4 |
| Median, 95% | 18.4 |

As the other testing parameters are the same as Table 2, the optimal amount of steps taken is still 18. From Table 3, it can be seen that the amount of steps taken is lower than that of Table 1. The average steps for all of the data would fail the test criterion, but the other three tests would pass, as they are within 20 steps. These results support the hypothesis made earlier.

EDC = -0.00000001

Figure 4: Learning curve for a lower EDC

Table 5: Averages and medians for a lower EDC

|  |  |
| --- | --- |
| Test Type | Steps Taken |
| Average, Full | 33.7 |
| Average, 95% | 28.2 |
| Median, Full | 28.2 |
| Median, 95% | 28.1 |

From Table 5, it can be seen that a lower EDC results in more steps taken for all tests when compared to the results from Table 2. It can then be concluded, from the results of Tables 2, 4, and 5, that a higher EDC can reduce the amount of steps taken to reach the goal. When a higher EDC, like the one used for Figure 3, is utilized, Test F is able to be completed.

**Conclusion**

The program is capable of creating an agent that understands the GridWorld in x,y coordinates. Tests D, E, and F are considered to be passed. Test G is incomplete, and as such, discussion is not possible for it. As such, the program is considered to be 75% complete, and the report is considered to be 75% complete. The results of the agent show that it is converging to a low value compared to its initial trials. Overall, the agent is particularly successful in learning when the GridWorld is represented as an x,y coordinate system.

**Disclaimer**

I worked with Matthew Jew on the project.

I used online resources to understand how to use 2D vectors and how to find the maximum value.

**Sources**

2D vectors - <http://stackoverflow.com/questions/1403150/how-do-you-dynamically-allocate-a-matrix>

Max values - http://www.cplusplus.com/forum/general/51452/