Britny Mors

ME 493 – Autonomy

5/11/2016

# Final Project: The Escape Artist

# Project Overview

The purpose of this project is to explore the use of the Q-learner algorithm in the application of a new domain. Previous projects (Project Beta) have explored the use of a Q-learner in a grid world domain as illustrated in figure 1. Here, an agent had a total of four possible actions for each state: North, South, East, and West. By implementing a Q-learning algorithm in conjunction with our agent we were able to navigate the Agent to a goal location using state information.

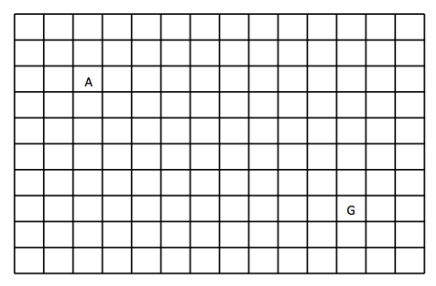
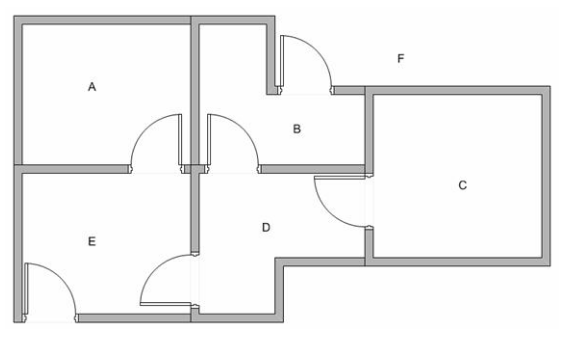


Figure : Grid world domain for Project Beta.

This project explores a variation to this problem by changing the domain in which the agent exists. For this project, a Q-learning algorithm was developed that navigates an agent throughout rooms of a house with a goal of finding the exit. This program was utilized in two different scenarios for this. First the agent navigated its way through a simplistic floor plan as show in figure 2 where the total number of states was six.



1

2

6

5

4

3

Figure : Simplistic floor plan consisting of a total of six states [1].

Next, the same algorithm was used to navigate an agent through a much more complicated floor plan which is shown in figure 3. This floorplan as opposed to the first floor plan only contains one exit and thirteen individual states.

1

2

3

4

5

6

7

8

9

10

11

12

13

Figure : Complicated floor plan with a total of 13 states.

The major difference between this project and the original project is each state has a varying amount of actions. Some rooms only have one door while others have up to three. In addition to this it is also much more difficult to track the agents current state as it cannot be mapped using the same methods utilized in project beta.

# Background

Q-learning is a reinforcement learning method that utilizes state information in order to determine actions that results in positive rewards. In order to fully utilize a Q-learner, both a reward table and a Q-table.

A reward table is utilized to provide positive rewards for desirable behavior and negative rewards for undesirable behavior. For example, figure 4 illustrates a simple reward table similar to that used in project beta where each state contains its own reward.



y

x

Figure : Reward table for a 4x4 grid

A Q-table contains data that presents information about the history of rewards and actions. A Q-table is built utilizing both states and actions with specific Q-values for a particular state or action. For example, in project beta a Q-table similar to that seen in figure 5 was utilized to store Q-values.

Figure :Q-table for a 2x2 matrix

In a Q-table, each value corresponds to a Q-value which can be calculated using equation 1 below.

In equation 1, alpha denotes the learning rate, gamma represents the discount factor, Q(s,a) is the old Q-value, R is the reward at the new state, and Qmax(s’,a’) is the max Q-value possible in the new state.

# Method

For this project the Q-table and reward table were structured slightly different than in project beta. In project beta, all actions were available for all states and each states Q-value corresponded directly to an action. For this project the Q-value instead was directly related to the new state.

For example, in project beta, selecting the action “North” in state one corresponds to the Q-value stored in the table at (1, 1). However, for the escape artist project, the Q-table was set up as seen in figure 6 below.

Figure : Q-table for simplistic floorplan.

Here each “action” is considered moving into the selected room. The -1000 values are Q-values that will never be updated because they are not plausible actions. For example, I cannot make it to room two from room one, therefor that Q-value can never update. Since the Q-value should never converge to -1000 we also know that certain actions will not be selected when choosing the greedy option during the decision process.

To prevent the Q-learner from selection an action that is not physically possible during the exploratory decision process the reward table was used. The reward table was formatted as seen in figure 7.

Figure :Reward table for simple floorplan.

In this reward table 100 indicates goal has been accomplished, -1 indicated possible but undesirable action, and 0 indicates impossible actions. For example, if the agent moves from room 2 to room 6 the agent exits the house and gains a reward of 100 and the corresponding Q-value for that action is updated. During the greedy selection process if an action corresponded to a reward of 0 a new action was selected until this was not true.

# Results

The developed Q-learning algorithm was able to be implemented in both scenarios with no issues. For the small floor plan 300 stat runs were generated with 1000 iterations during each stat run. Each time the Agent was able to escape the house. A learning curve was developed to determine how long it typically took the agent to escape the house. Figure 8 below shows the learning curve for the small floor plan.

Figure : Simple floor plan learning curve.

This learning curve shows that the agent was able to figure out how to exit the house in two moves incredibly quickly. This is expected as the small floor plan is incredibly simple and each room can be escaped in a max of two steps.

Next the Q-learning algorithm was implemented using the large floor plan. Figure 9 shows the learning curve after 30 stat runs of 1000 iterations each for the large floor plan.

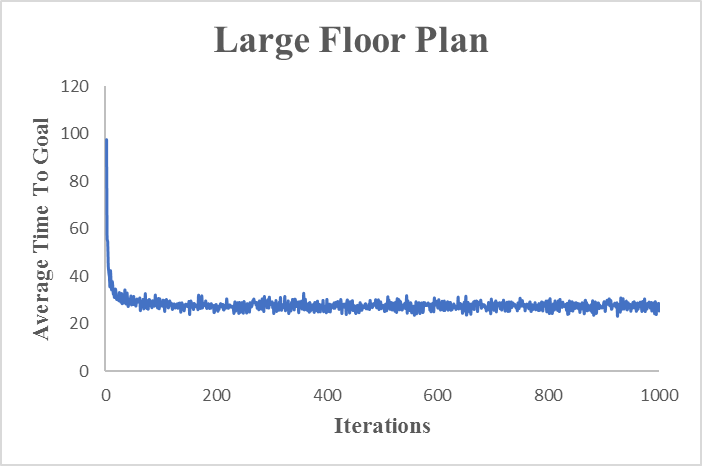


Figure : Large floorplan learning curve.

This shows that the learner took a lot longer to exit the house which is to be expected with a much larger and more complex floor plan.

# Conclusion

Overall the results from this project were favorable. Although the Q-learner takes a longer amount of time to escape the larger house it does in fact learn an efficient method of doing so. This would be interested to take the problem to an in fact larger and more complicated floor plan. This would be difficult with the current method however because the larger the floor plan the larger the reward table which was coded by hand due to the complexity of the layouts. Another interesting thing to look at would be how to reduce the number of states to avoid state space explosion given a larger floor plan.