UDACITY

Machine Learning Nanodegree
Plot and Navigate a Virtual Maze
Capstone Project Report

Prepared by: Clive R. Cadogan

Sunday, October 29, 2017

I. Definition

Project Overview

This project is based on Artificial Intelligence Path Planning. Robot navigation is a generalization of the route-finding problem This project takes inspiration from Micromouse competitions, wherein a robot mouse is tasked with plotting a path from a corner of the maze to its center. The robot mouse may make multiple runs in a given maze. In the first run, the robot mouse tries to map out the maze to not only find the center, but also figure out the best paths to the center. In subsequent runs, the robot mouse attempts to reach the center in the fastest time possible, using what it has previously learned.

The approach may also be applicable to path planning for self driving cars that need to plan the optimum route to the desired destination and other planning challenges.

Problem Statement

To search and determine the best path towards a desired location. This project will utilize a Reinforcement Leaning based algorithm than can help a robot navigate the fastest way through a maze. This may be useful solving path planning problems such as in self driving cars and other real world robots that need to navigate the real world to accomplish a goal.

The robot will first explore the maze tho collect data that may be used to compute the optimum path. Once the optimum path is determined by the algorithm the robot must be able achieve the goal within the required time limit.

Metrics

The score received for successfully completing the game based on number of time steps required to execute the second run, plus one thirtieth the number of time steps required to execute the first run.

II. Analysis

Data Exploration and Visualization

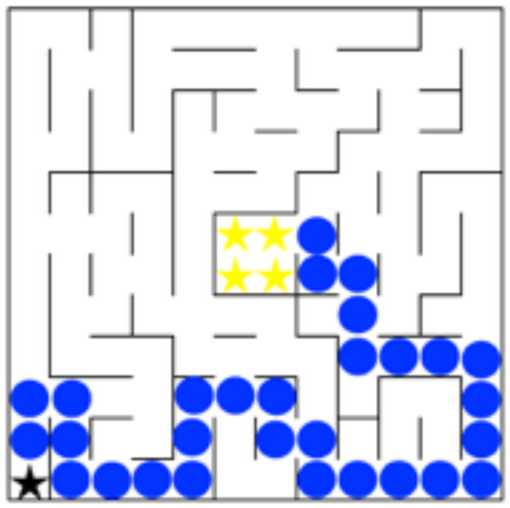
All the mazes are squares and the start location for the agent is always at the bottom left corners (0, 0). In the center of the grid is the goal room consisting of a 2 x 2 square.

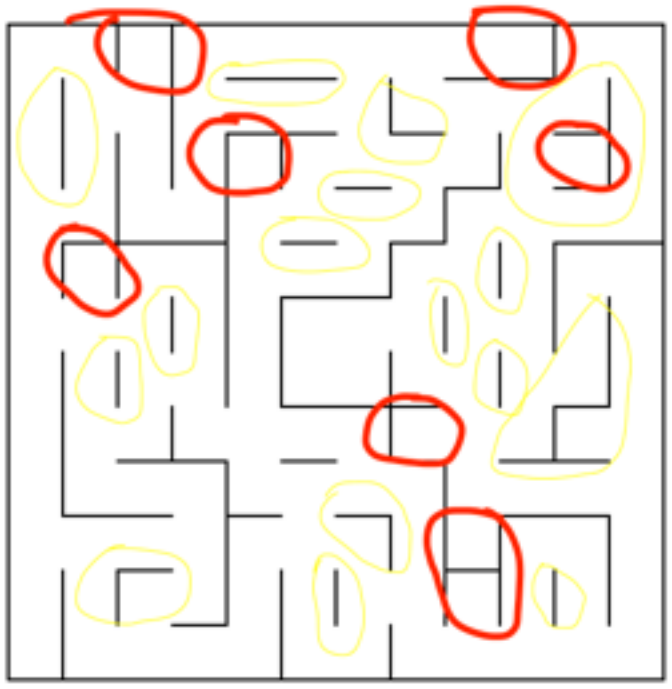
The robot has three obstacle sensors, mounted on the front of the robot, its right side, and its left side. Obstacle sensors detect the number of open squares in the direction of the sensor.

It was decided to limit the robots movement to one step if possible for each time step. During each time step the robot may choose to rotate +/-90 degrees then move forward one step.

It is assumed that the sensors have 100% accuracy however, if the step tries to take the robot into a wall the robot stays there.

Maze No. 1 (12X12)
The shortest path to the goal for maze one is 30 single steps. Despite the robot being able to take a maximum of three steps it was decided to utilize only steps of one in the desired direction once permitted by absence of a blocking wall.





Maze No. 1 Optimum path to goal

Maze No. 1 Things to avoid - Dead-ends and loops

The robot will learn a policy to avoid dead-ends by being aware a penalty whenever one is reached during the first run. Hence these locations will be avoided during the second run by utilizing the optimal policy by simply selecting the action with the highest value in each state to navigate through the maze to the goal state.

The robot will learn a policy to find the optimal path and avoid loops by being awarded increasing rewards for progressively moving towards the goal during the first run. Hence loops will be avoided during the second run by utilizing the optimal policy by simply selecting the action with the highest value in each state to navigate through the maze to the goal state.

Algorithms and Techniques

An a Reinforcement Learning algorithm will be utilized to identify and plot the most efficient path towards the desired goal.

The task of reinforcement learning is to use observed rewards to learn an optimal (or nearly optimal) policy for the environment[1].

[1] Russell, Stuart J., and Peter Norvig. Artificial intelligence: a modern approach. Pearson, 2016.

The approach that will be utilized is Q-learning which is a model-free reinforcement learning technique. It works by learning an action-value function that ultimately gives the expected utility of taking a given action in a given state and following the optimal policy thereafter. A policy is a rule that the agent follows in selecting actions, given the state it is in. When such an action-value function is learned, the optimal policy can be constructed by simply selecting the action with the highest value in each state. One of the strengths of Q-learning is that it is able to compare the expected utility of the available actions without requiring a model of the environment[2].

The agent will constructs a Q-table that will represents the learned action-value function as traverses the the maze in the first run. Then during the second run the agent will utilize the optimal policy by simply selecting the action with the highest value in each state to navigate through the maze to the goal state.

Benchmark

	Maze No. 1	Maze No. 2	Maze No. 3
Estimate for Run No. 1 (N^2/30)	4.8	6.5	8.5
Estimate for Run No. 2	30	40	47
Estimated Score	34.8	46.5	55.5

[2] "Q-Learning." Wikipedia, Wikimedia Foundation, 20 Oct. 2017, en.wikipedia.org/wiki/Q-learning.

III. Methodology

Data Preprocessing

This project did not require any data preprocessing since the sensor specifications and environment designs were provided as part of the starter package.

Implementation

1. Rewards and penalties were set up in order to have the robot learn an optimal policy to get to the goal.
2. Increasing rewards were give at each increasing from -1 at the outermost bounds to 1000 at the goal.
3. Penalties was awarded for when the agent found it self in a dead-end to her learn and optimal policy to avoid dead-ends.
4. Valid actions are actions are determined based on the sensor readings, each action was represented a tuple of two values the first is the rotation and the second is the movement. NB. Movements were limited to one step to ensure the option at each locations are assessed identify the best.
5. The Q-table is a dictionary of the states visited by the agent and the actions taken from each state and the calculated value of those actions.
6. The state object used in the Q table was a six element tuple made up of the three values from the sensors, the heading and location to ensure each state could be uniquely identified in order store corresponding actions taken and rewards or penalties awarded in value faction's Q table.
7. During the first run some actions are chosen randomly to allow explorations and some actions are chosen based on the learned optimal policy at that point in time.
8. During the second run actions are only chosen based on the learned optimal policy in order to get agent to the goal as quickly as possible.

Refinement

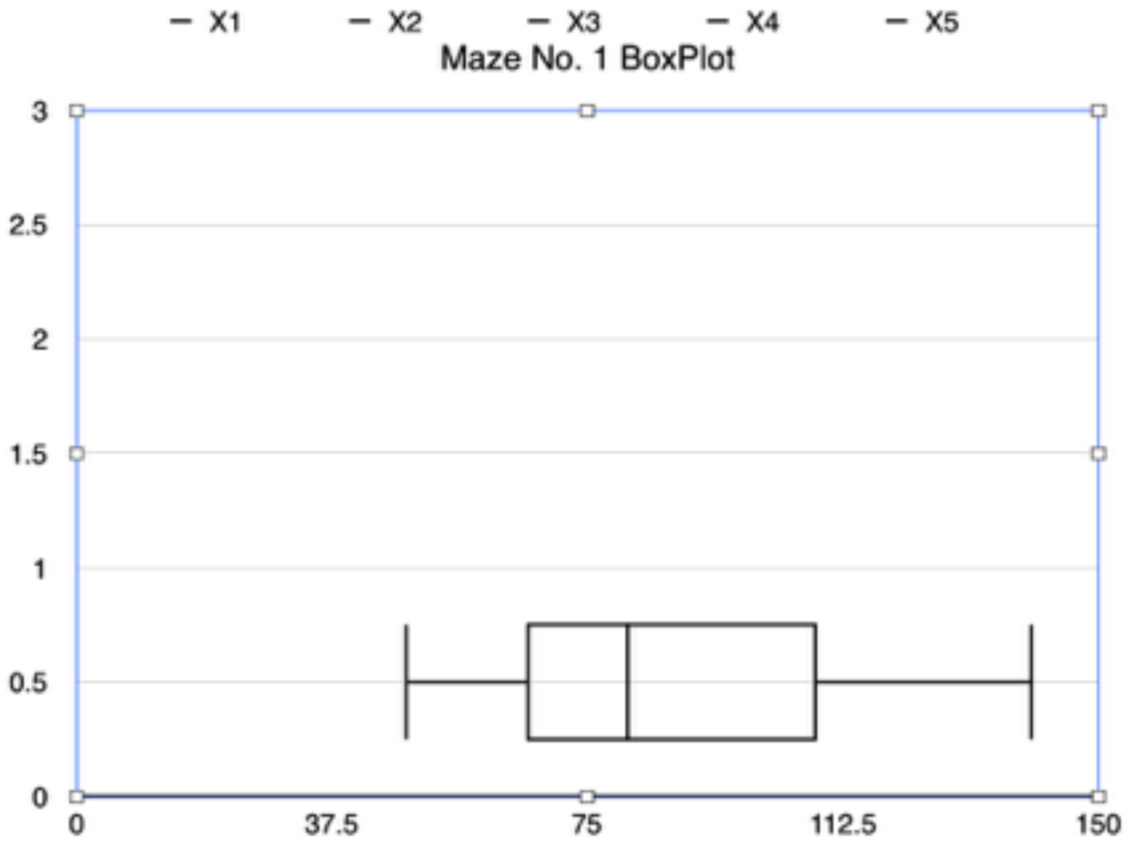
Initially the steps the robot took was allowed to vary from one the the max of what was possible. At this point the the robots ability to complete the maze successfully was highly in consistent and the time take was much higher. The robots performance was much more robust when the steps were limited to one step.

Ideally to learn an optimal policy for the agent to efficiently navigate to the goal state the agent should explore all the possible routes from the starting point to the goal. However due to being only limited to one training run the original implementation didn't restart from the origin but attempted explore as much of the maze as possible in one go. This lead to far below optimal results. Subsequently a function implemented allow the agent to navigate back to the starting point after reaching the goal to allow the opportunity to find more effective paths to the goal state. This improved the times on the second run significantly.

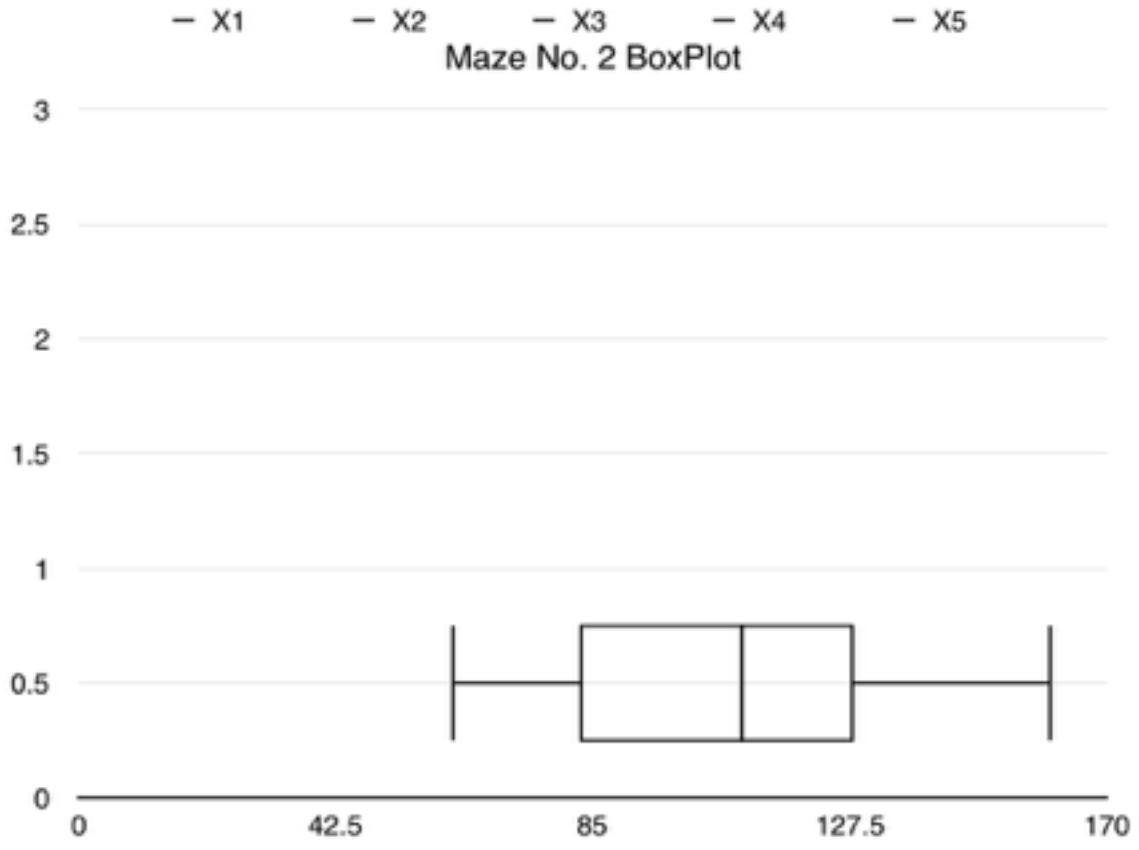
IV. Results

Model Evaluation and Validation

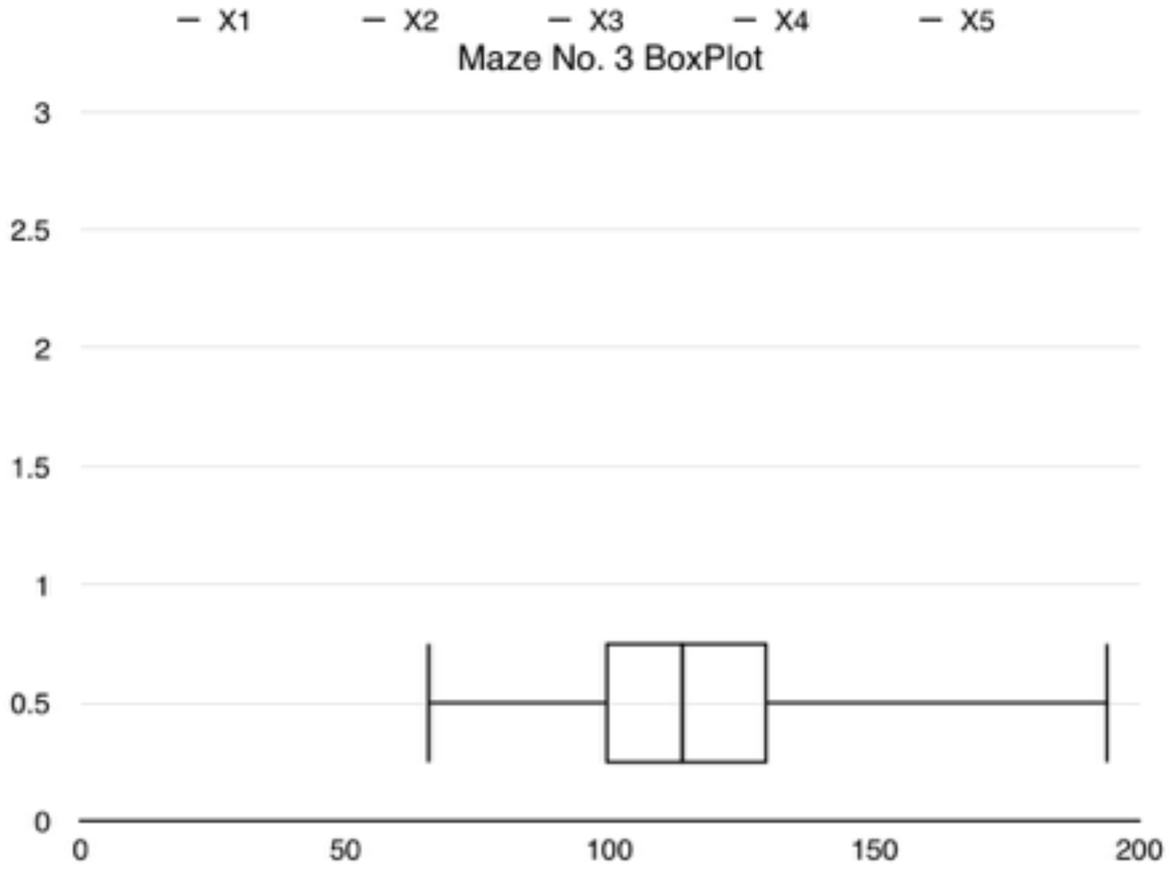
The results for agent on Maze No. 1 were much higher than what would be considered optimal as can be see by the box plot below of the results. The results display a wide variation with a minimum value of 48 to maximum of 140 with distribution positively skewed towards its lower bound and the average score was twice the estimate. This may be due to the limited training runs/time.



The results for the agent on Maze No. 2 were negatively skewed with a minimum of 62 and max of 160 and average score was approximately twice the estimate. Once again this is believed to be due to the limited about of training runs due to the limited time.



The results for agent on Maze No. 3 were much higher than what would be considered optimal as can be see by the box plot below of the results. The results display a wide variation with a minimum value of 65 to maximum of 329 and the average score was twice the estimate. This may be due to the limited training runs/ time. The standard deviation for the results on this maze was almost that of the others, possibly indicating a decrease in robustness for larger mazes given the exploration constraints.



Justification

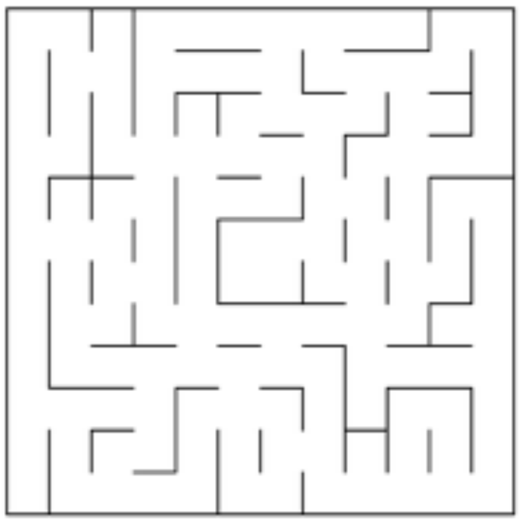
The results for the three mazes were approximately twice there benchmarks. The results seem to become less robust with larger mazes based on the apparent logarithmic increase in standard deviation with size of the mazes. This may be due to inadequate time to restart exploration of the maze which is required to find an optimal policy.

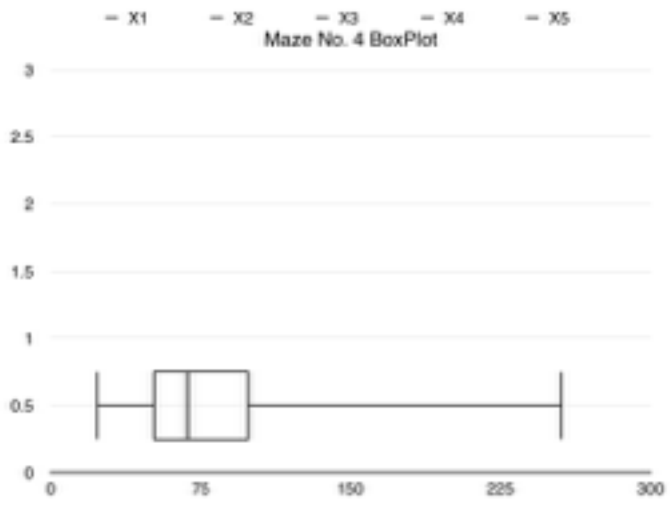
	Maze No. 1	**Maze No. 2**	**Maze No. 3**
Estimate for Run No. 1 (2N^2/30)	9.6	13	17
Estimate for Run No. 2	30	40	47
Estimated Score	39.6	53	66
Average of Results	86	109	127
Standard Deviation	25	26	49
Min. Results	48	62	65
Max. Result	140	161	329
Q1	68	84	104
Q2	80	110	120
Q3	108	127	140

V. Conclusion

Free-Form Visualization

1. This maze was designed with more opportunity to access the goal with the hope that it confirm that due to limited restart runs due to time constraints the results of the agent may not be as good as they could be.
2. Based on the box plot of the resulting data for this maze it is a strong possibility that this algorithm needed more training from the starting point of the maze to learn an optimal policy in order to more consistently reach the in a more optimal manner.
3. This maze presented the agent with more opportunities/paths to get to the goal but the results was still widely varying.





Reflection

1. It was decided to use the Q learning reinforcement learning approach based on my confidence in it after implementing a similar solution as part of the nanodegree.
2. Rewards were setup based on location in a contoured heat map radiating from the goal with goal have the highest reward value and descending the further the agent was away from the goal. Penalties were aware for dead-end locations and corners with dead-ends incurring the highest penalty.

3. During the exploration phase the next actions are chosen randomly based on a list of possible actions that may or may not be filtered for highest rewards.
4. After an action is taken the agent updates its location and heading parameters for the next iteration and a reward is computed for the taken action value function/Q-table is updated based on the previous state, the action taken and the computed reward.
5. During the second run the agent identifies the optimum learned polity from the Q-table and uses it to navigate its way to the goal as best it can.
6. It was required to implement a few additional things due to constraints laid out for this project such as only one trying run.
7. Ideally the chosen approach needs to have the agent explore all possible paths starting each time for the starting point in order to learn an optimal policy. Due the limited time it was very difficult to get the required restarts to explore all paths from the begin.

Improvement

1. To ensure the reinforcement approach can find the optimal policy the agent should be allowed to navigate the environment and collect adequate data as required to learn an optimal policy.
2. To be used in a more real world environment a extensive localization and mapping system would be required to compensate to variations in measurements, errors movements and changes in the environment.
3. A more suitable programming language to interface with real hardware may also be required.
4. Additional research would also be needed to identify or develop a more comprehensive approach to reinforcement learning.