|  |  |
| --- | --- |
| Capstone Project  Machine Learning Engineer Nanodegree | Naoki Shibuya  March 12, 2016 |

Plot and Navigate a Virtual Maze

# Definition

## Project Overview

This project takes inspiration from the micro mouse 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.

In this project, I will create functions to control a virtual robot to navigate a virtual maze. A simplified model of the world is provided along with specifications for the maze and robot; my goal is to obtain the fastest times possible in a series of test mazes.

## Problem Statement

I would like to try different implementation of goal seeking behaviors. Therefore, I am going to develop a framework where I can use different robot controlling logic/implementation for measuring the performance. This way, I can develop a number of robot controller programs that can navigate in a virtual maze in its own way. I will be improving the performance of the robot as I progress further in this project.

## Metrics

On each maze, the robot must complete two runs. In the first run, the robot is allowed to freely roam the maze to build a map of the maze. It must enter the goal room at some point during its exploration, but is free to continue exploring the maze after finding the goal. After entering the goal room, the robot may choose to end its exploration at any time. The robot is then moved back to the starting position and orientation for its second run. Its objective now is to go from the start position to the goal room in the fastest time possible. The robot’s score for the maze is equal to the number of time steps required to execute the second run, plus one thirtieth the number of time steps required to execute the first run. A maximum of one thousand time steps are allotted to complete both runs for a single maze.

# Analysis

## Data Exploration

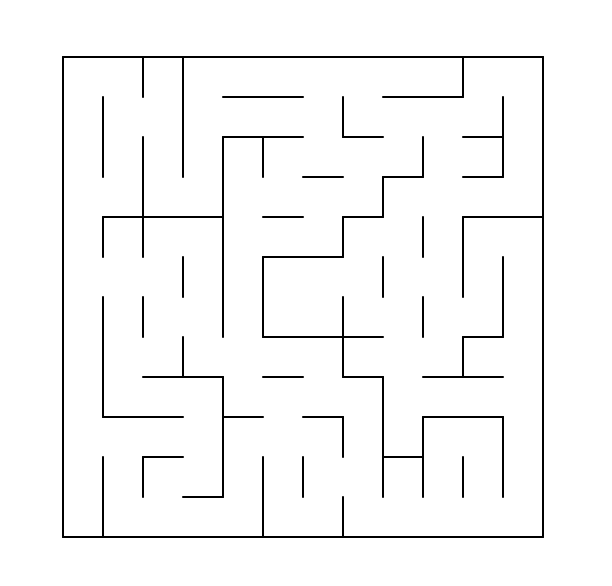
The robot can be considered to rest in the center of the square it is currently located in, and points in one of the cardinal directions of the maze. The robot has three obstacle sensors, mounted on the front of the robot, its right side, and its left side. Obstacle sensors detect the number of open squares in the direction of the sensor; for example, in its starting position, the robot’s left and right sensors will state that there are no open squares in those directions and at least one square towards its front. On each time step of the simulation, the robot may choose to rotate clockwise or counterclockwise ninety degrees, then move forwards or backwards a distance of up to three units. It is assumed that the robot’s turning and movement is perfect. If the robot tries to move into a wall, the robot stays where it is. After movement, one time-step has passed, and the sensors return readings for the open squares in the robot’s new location and/or orientation to start the next time unit.

More technically, at the start of a time step the robot will receive sensor readings as a list of three numbers indicating the number of open squares in front of the left, center, and right sensors (in that order) to its “next\_move” function. The “next\_move” function must then return two values indicating the robot’s rotation and movement on that time step. Rotation is expected to be an integer taking one of three values: -90, 90, or 0, indicating a counterclockwise, clockwise, or no rotation, respectively. Movement follows rotation, and is expected to be an integer in the range [-3, 3] inclusive. The robot will attempt to move that many squares forward (positive) or backwards (negative), stopping movement if it encounters a wall.

Any maze in this project is a square. The first maze is 12x12. The second is 14x14. The third is 16x16. The bigger maze should be more difficult to search the goal as there are more ways to reach it.

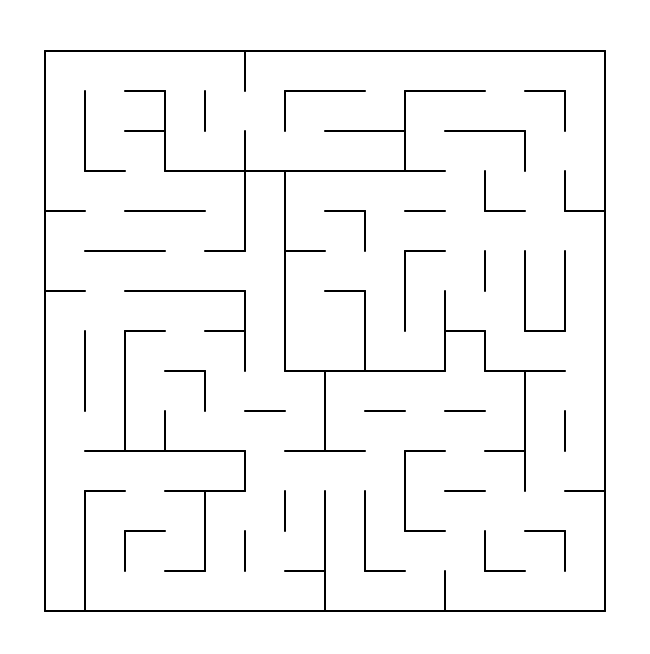
## Exploratory Visualization

### Maze 01 (12x12)

The optimal path in the first maze takes 30 steps to reach the goal. The challenge in this maze is that there are many potential loops. Also, there are several dead-ends.

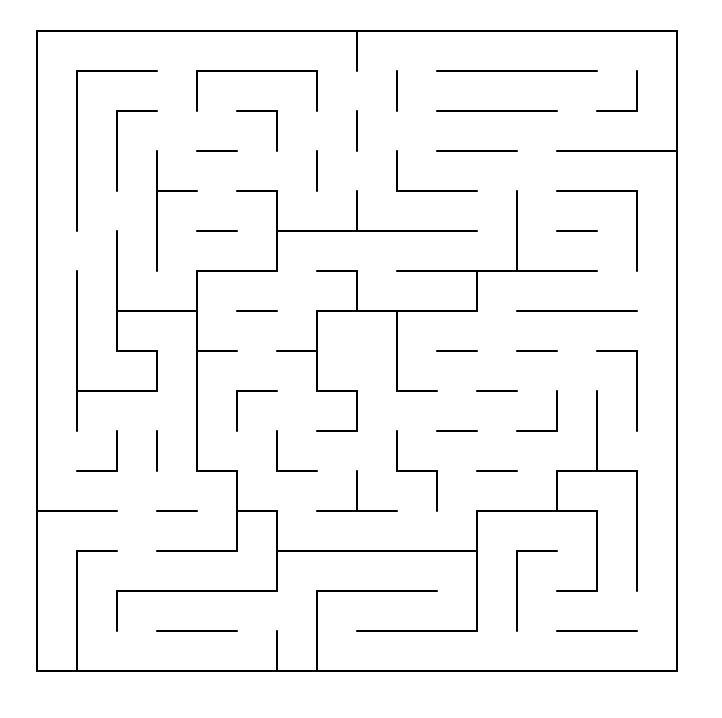
### Maze 02 (14x14)

It is not easy to see where is the shortest path to the goal. There are more dead-ends than the maze 01. There are many loops, too.



### Maze 03 (16x16)

Less dead-ends than the other two but they seems to have longer path to it. So, if we do not avoid the dead-ends, we will be wasting lots of times there. Many potential mini loops are there too.



## Algorithms and Techniques

Various search algorithms will be used in this projects. First of all, I will develop a robot controller that has no memory or smartness. In fact, it will behave randomly. The performance of such controller will set the base line benchmark for the subsequent controllers. I will introduce heuristic value function and then dynamic programming to update location to value mapping dynamically.

## Benchmark

I will run each robot controller for 100 times to measure the average score and the standard deviation so that I can see the performance of each controller implementation and also compare them to see the improvements. The performance should be measured against different mazes as well so that we can compare how the same controller behaves in different mazes.

# Methodology

## Data Pre-processing

The maze specification and robot's sensor data is provided and 100% accurate. Therefore, there is no data pre-processing is required.

## Implementation

I will develop several different robot controllers so that I can see the performance differences.

The first robot controller will be used to set a baseline performance. It will behave randomly and have no memory or smartness except that it is aware of walls by reading the sensor values. When it reaches to a dead-end, it will randomly turn left or right. However, it will not remember the dead-end so it may come back to it again and again. I expect this controller will fail to reach the goal most of the times.

The second robot controller will have memory of where it has been and avoid going into dead-ends. Otherwise, it is still moving randomly. It should be more likely to reach the goal than the first one as less area will be used to explore once dead-ends are found. It may still looping around and going back to the same path over and over.

The third robot controller will keep track of how often it visit each location in a maze so that it can explore into less often visited locations or paths. This should avoid looping the same path over and over as it needs to move towards less explored areas eventually. I expect this controller to reach the goal much more often than the first and the second ones. But it would be slow to reach the goal as it is not aware of where the goal is. Also, this controller will be able to exploit what was explored from the first run.

The fourth robot controller will be using a heuristic function so that it can choose a path that is potentially closer to the goal. All the good features from the first, second and third controllers will be included. It should be faster than the third robot to reach the goal. However the heuristic function calculates the distance to the goal assuming no walls. So, it is not always correct measure of the distance to the goal and it may push the robot into a loop (which should be avoided by choosing less visited location/path).

The fifth robot controller will be dynamically adjust the heuristic function when the walls prevent the robot to move to a location with smaller heuristic value than that of the current location. This robot should be even faster than the previous one. It should be able to move backwards from a dead-end so that it is faster to move out from the dead-end than turning twice.

The sixth robot controller will be able to take up to 3 steps both forward and backward. So, that it can move faster to reach the goal than the fifth robot, especially in the second run.

## Refinement

In this section, you will need to discuss the process of improvement you made upon the algorithms and techniques you used in your implementation. For example, adjusting parameters for certain models to acquire improved solutions would fall under the refinement category. Your initial and final solutions should be reported, as well as any significant intermediate results as necessary. Questions to ask yourself when writing this section:

Has an initial solution been found and clearly reported?

Is the process of improvement clearly documented, such as what techniques were used?

Are intermediate and final solutions clearly reported as the process is improved?

# Results

## Model Evaluation and Validation

In this section, the final model and any supporting qualities should be evaluated in detail. It should be clear how the final model was derived and why this model was chosen. In addition, some type of analysis should be used to validate the robustness of this model and its solution, such as manipulating the input data or environment to see how the model’s solution is affected (this is called sensitivity analysis). Questions to ask yourself when writing this section:

Is the final model reasonable and aligning with solution expectations? Are the final parameters of the model appropriate?

Has the final model been tested with various inputs to evaluate whether the model generalizes well to unseen data?

Is the model robust enough for the problem? Do small perturbations (changes) in training data or the input space greatly affect the results?

Can results found from the model be trusted?

## Justification

In this section, your model’s final solution and its results should be compared to the benchmark you established earlier in the project using some type of statistical analysis. You should also justify whether these results and the solution are significant enough to have solved the problem posed in the project. Questions to ask yourself when writing this section:

Are the final results found stronger than the benchmark result reported earlier?

Have you thoroughly analyzed and discussed the final solution?

Is the final solution significant enough to have solved the problem?

# Conclusion

## Free-Form Visualization

Use this section to come up with your own maze. Your maze should have the same dimensions (12x12, 14x14, or 16x16) and have the goal and starting positions in the same locations as the three example mazes (you can use test\_maze\_01.txt as a template). Try to make a design that you feel may either reflect the robustness of your robot’s algorithm, or amplify a potential issue with the approach you used in your robot implementation. Provide a small discussion of the maze as well.

## Reflection

In this section, you will summarize the entire end-to-end problem solution and discuss one or two particular aspects of the project you found interesting or difficult. You are expected to reflect on the project as a whole to show that you have a firm understanding of the entire process employed in your work. Questions to ask yourself when writing this section:

Have you thoroughly summarized the entire process you used for this project?

Were there any interesting aspects of the project?

Were there any difficult aspects of the project?

Does the final model and solution fit your expectations for the problem, and should it be used in a general setting to solve these types of problems?

## Improvement

Consider if the scenario took place in a continuous domain. For example, each square has a unit length, walls are 0.1 units thick, and the robot is a circle of diameter 0.4 units. What modifications might be necessary to your robot’s code to handle the added complexity? Are there types of mazes in the continuous domain that could not be solved in the discrete domain? If you have ideas for other extensions to the current project, describe and discuss them here.

## Before submitting your report, ask yourself…

Does the project report you’ve written follow a well-organized structure similar to that of the project template?

Is each section (particularly Analysis and Methodology) written in a clear, concise and specific fashion? Are there any ambiguous terms or phrases that need clarification?

Would the intended audience of your project be able to understand your analysis, methods, and results?

Have you properly proof-read your project report to assure there are minimal grammatical and spelling mistakes?

Are all the resources used for this project correctly cited and referenced?

Is the code that implements your solution easily readable and properly commented?

Does the code execute without error and produce results similar to those reported?