Machine Learning Nanodegree

Capstone Project

David lee

Project overview

definition

The project is aimed at using a robot mouse tasked with plotting a path from a corner of the maze to its center that is use a robot mouse to navigate through unknown maze learn the path from initial position to the goal area in an exploration run using its sensors and used this acquired knowledge and to navigate again from this initial position to the goal(center) using the optimal path(shortest path) taking shorter steps and least time possible. This project takes inspiration from Micromouse competitions in This video.

Problem statement.

The robot mouse must get to the goal(center) area of the maze,using the optimal path(shortest path),reduced steps and shortest time possible. Also on each maze, the robot must complete two runs,the first run called exploration run the robot is allow explore all the necessary cells and path from initial position left corner thats that leads to the goal using its sensors for detection of obstacle which means when its meets a wall its stops movement and rotates to take the next movement and store information while mapping the path(wall map). Once its gets to the center which is a 2x2 square cells,its is free to continue exploring the maze after finding the goal and may choose to end its exploration at any time and then reset to return to the initial pose do an exploitation run with the acquired knowledge from first run using the optimal path.

Metrics

The performance metric used for evaluation is the score since its take into consideration all runs the robot made that is on each maze, the robot must complete two runs ,in the first run each time step score 1/30point of the points and for the second run each time step score a full point. A maximum of one thousand time steps are allotted to complete both runs for a single maze. An extra step can be advantageous in the first run to minimize the number of steps. The robot’s score for the maze is equal to the number of time steps required to execute first run plus the number of time steps required to

execute second run. This score for the robot implies that the robot had left the initial pose,left corner of the maze and reached final pose (goal),center of the maze.

Analysis and Visualization

Mazes are provided to the system via text file which is constructed as nxn square cells,n is even . Each number represent the presence or absence of wall .On the first line of the text file is a number describing the number of squares on each dimension of the maze n. Each number represents a four-bit number that has a bit value of 0 if an edge is closed (walled) and 1 if an edge is open (no wall); the 1s register corresponds with the upwards-facing side, the 2s register the right side, the 4s register the bottom side, and the 8s register the left side. For example, the number 10 means that a square is open on the left and right, with walls on top and bottom (0*1 + 1*2 + 0*4 + 1*8 = 10).Bellow is a representation of 12x12 square cells from text file, on the first line of the text file is a number describing the number of squares on each dimension of the maze.

```
12
1,5,7,5,5,5,7,5,7,5,5,6
3,5,14,3,7,5,15,4,9,5,7,12
11,6,10,10,9,7,13,6,3,5,13,4
10,9,13,12,3,13,5,12,9,5,7,6
9,5,6,3,15,5,5,7,7,4,10,10
3,5,15,14,10,3,6,10,11,6,10,10
9,7,12,11,12,9,14,9,14,11,13,14
3,13,5,12,2,3,13,6,9,14,3,14
11,4,1,7,15,13,7,13,6,9,14,10
11,5,6,10,9,7,13,5,15,7,14,8
11,5,12,10,2,9,5,6,10,8,9,6
9,5,5,13,13,5,5,12,9,5,5,12
```

These values are unknown to the robot in the beginning but as the robot start to make its way through the maze to build a grid using these values and the information sensed by the sensor its becomes known to the robot. due to array indexing, the first data row in the text file corresponds with the leftmost column in the maze, its first element being the starting square (bottom-left) corner of the maze. This imply that 1 in the first column and first row in the text file above represents the starting position in the bottom left corner in the maze. Below is the visualization of the maze constructed from text file by writing python showmaze.py test_maze_01.txt on the command line.

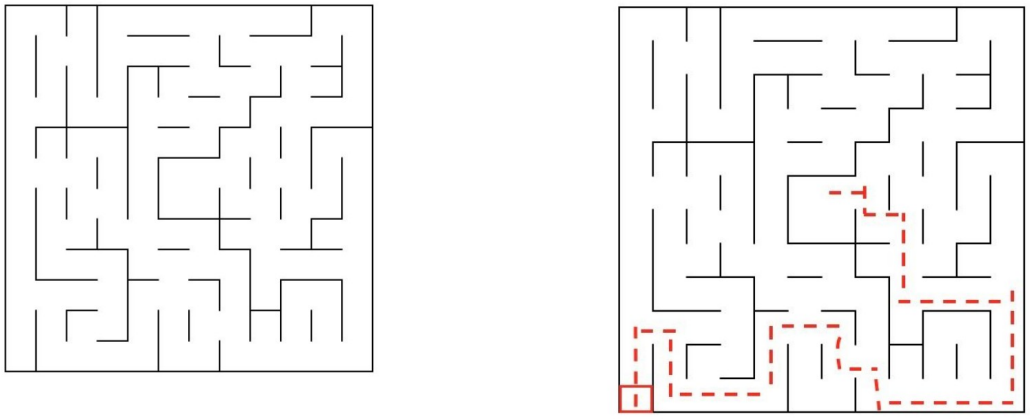

Fig 1 (a) :represent a produced maze

Fig (b):represent the shortest path used by the robot.

From this visualization its can be deduced that due to the fact that the robot has sensors for detection and revealed information about direction from the environment, therefore for the start the left and right sensors will read 0 because they are at the side and the sides are closed walls while the front sensor will read 1 to the top which shows that the front is an opening. Each cell represent a number of time step the robot can take.

Algorithms and Techniques

There are several algorithm for finding the shortest path through a maze both the one in which the maze is known to the robot and that which the maze is unknown to the robot,here,the maze is unknown to the robot and the robot has to learn it by itself. Therefore the first step is to get an algorithm that will find ,discover and map all the possible paths that leads to goal minimizing the steps so that in the subsiquent run the robot can use this path as shortest path to the goal. The algorithm used to do this is A*algorithm is used for path finding even though not optimal path and graph traversal. A* is an informed search algorithm, or a best-first search, meaning that it solves problems by searching among all possible paths to the solution (goal) for the one that incurs the smallest cost (least distance traveled, shortest time, etc.), and among these paths it first considers the ones that *appear* to lead most quickly to the solution. When given a starting position and the target position,A* progressively search through all possible paths of length increasing the length through the search space until the target is found. This algorithm finds the path to the gaol but not optimal path. Secondly ,another algorithm is used to find the the shortest path to the goal called the optimal path. The algorithm used to do this is Dijkstra's algorithm which is an algorithm for finding the shortest paths between nodes in a graph .For a given source node

in the graph, the algorithm finds the shortest path between that node and every other. It can also be used for finding the shortest paths from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined.

With this algorithm in place and the information obtain from the sensors of the robot ,the robot will be able to keep track of the number of cells and the shape of the maze which will enable it accomplish the task of navigating from its initial position to the final (goal) position.

Benchmark

Also for each maze the robot have to execute two runs, the first run called exploration run in which this gives its the opportunity to learn the environment,explore every path randomly to build a map of the maze from the initial pose to the goal pose and other path presents so as to achieve our goal we interested in in the second run efficiently.. During the second run called the exploitation run where the robot must have had full knowledge of its environment and must maximize this knowledge where its takes the shortest time and the best shortest path with reduced number of steps to get to the target called optimal path to the target and for the first maze that is the 12x12 square maze require a to take 17 steps for the optimal path. After this exploration exploitation run the robot is awarded a score which in this case it is used as the performance metrics. This reward score is given as 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 given by

Score = Nsr + 1/30Nfr

Nsr = number of steps for the second run,

Nfr = number of steps for first run.

Methodology

Data Preprocessing

With the information on the specification of the robot mouse and the maze given ,therefore data preprocessing is not necessary.

Implementation.

To implement work, some steps(work flow as seen in the proposal) are followed which help the code to flow down and also with the algorithm in it. To begin coding, import of some packages or libraries such as numpy and others which will be seen in navigator.py,a class function which begins by the use of initialization function to set up attributes the robot will use to learn and navigate the maze, initial state for the two-dimensional list that represents all of the walls in the maze including exterior walls and the number of possible shapes each cell could take.. Some initial attributes are provided based on some information including the size of the maze the robot is placed in such as location,heading etc. With the sensors in place to Save the latest sensor readings and the mapper to update its knowledge about the maze walls by Updating map of walls and openings based on current location, heading, and sensor readings.,also to decide the rotation and movement the robot should make in order to maximize knowledge of the maze. In order to improve the efficiency of the Robot's traversal of the maze and to avoid the robot getting stuck between two nodes indefinitely, the Navigator will attempt to follow the first two steps of each calculated path through the maze. The only cases when this isn't done is when the path has only one step or when the second step becomes invalid as a result of the sensor readings following the first step. To fine uncertainties,we find the node with the greatest uncertainty (greatest number of possible shapes) that is closest to the current location. In the course of navigating sot hat the algorithm will be used the maze map is converted to graph,the robot Will be moving from location to target, given the current knowledge of the maze. An algorithm comes in to get best path through graph,use Dijkstra's algorithm to find the fastest path from start to target through the given undirected graph. Determine whether the optimal path through the maze has been found. If this is the first time the optimal path has been found, save it. Checking the goal,the robot check to see if the goal entrance has been discovered. If either the goal opening has been found or all 7 goal walls have been found then the remaining wall locations or opening can be inferred respectively,save it reset and return for the second run and use best optimal path which is shortest from start to the goal with the score. To accomplish all this files such as mapper.py, navigator.py, maz_generator.py, and bach_maze_runner.py has been implemented together with the modified robot.py accompany tester.py to accomplish its task to run to test the robot’s ability to navigate mazes by writing python tester.py test_maze_01.txt on the command line which gets the specification of each maze from test_maze_##.txt and get the robot to navigate the maze while the command python showmaze.py test_maze_01.txt poduces the maze.

Refinement

An algorithm has been used to find the optimal path for the robot,also the code written which used the algorithm find this optimal path to the goal. So every cell in the in the maze visited has mapped out to produce a wall map,although some cells are unvisited but there is still a guarantee that the optimal path will be found since its is not a function of all the cells in the maze must be visited. With the code the robot can get its initial state,heading direction and the next move on how to go about with each run. There has been a challenge with large maze such as 16x16 where the robot struggle to explore and map out the entire maze and visiting dead end cells and getting stock, using algorithm to find the optimal path with many cells ,the algorithm may turn to fail because the its suppose to have a knowledge of the shape of the cells and sizes of cells in the maze for it to make a decision and the robot fail to enter the target. To improve this multiple visit to dead end cells should be avoided and not getting current caught traveling in loops in a maze and some codes to check this as seen in *navigator.py* in *class def check_for_repeats_in_visited_nodes(self):*.

Results

Model Evaluation and Validation

To efficiently evaluate the performance of the robot navigation and mapping logic on the 3 mazes and given the all the input data set to solve the problem,with the mouse specification in pace. To achieve this we use the relationship between the robot mouse and the maze sizes with the required number of steps to start from the initial pose to the final pose,goal which show a correlation as we will see below with the scores as expected and also the success of the algorithm for the 3 different mazes which show robustness and is replicably. The algorithm is consistent in finding the optimal path to the goal since it hasn’t fail in its execution to for finding optimal path from the initial pose to goal pose for the given 3 different mazes,therefore this makes it a robust algorithm. Any change in maze size does not affect it and promising that its can handle maze with size greater than this. Hence model trusted. Here is the results ;

Maze 1

Best found path: [(0, 2), (1, 2), (1, 0), (4, 0), (4, 2), (6, 2), (6, 1), (7, 1), (7, 0), (10, 0), (11, 0), (11, 3), (8, 3), (8, 5), (7, 5), (7, 6), (6, 6)]

Best found path steps count: 17

Task complete! Score: 20.667

Maze 2

Best found path: [(0, 3), (2, 3), (2, 2), (1, 2), (1, 0), (4, 0), (5, 0), (5, 3), (8, 3), (8, 1), (10, 1), (10, 2), (12, 2), (12, 5), (13, 5), (13, 8), (13, 9), (10, 9), (8, 9), (8, 8), (6, 8), (6, 7)]

Best found path steps count: 22

Task complete! Score: 28.567

Maze 3

Best found path: [(0, 3), (2, 3), (2, 4), (0, 4), (0, 7), (0, 10), (0, 13), (0, 15), (3, 15), (6, 15), (7, 15), (7, 12), (8, 12), (8, 11), (11, 11), (11, 10), (8, 10), (6, 10), (6, 8), (5, 8), (5, 7), (6, 7), (6, 5), (8, 5), (8, 7)]

Best found path steps count: 25

Task complete! Score: 31.767

Justification

Here,for the results to be justified the the first parameter to use to justify is the benchmark as earlier sets and in which its has been made as expected from robot on the maze in other to find optimal path from initial pose to final pose goal. Also to complete two runs with the first run and exploratory run to gain knowledge about the maze and maximize in the second run exploitation run using optimal path with reduced steps from the first run and shortest time achieving good scores. The performance is not more then expected or less,I did set just the standard and if its were not made before I could go into checking and setting other parameters. Therefore its just solve the problem of virtual mouse in the maze.

Conclusion

Free-Form Visualization

Here,will look at other cases such as a maze with more dead ends and some potential loops that the robot may get stock in making its difficult for the robot to reach the goal using its optimal path. With the model formal used if the robot can navigate through the maze correctly in a maze of 12x12 getting to the goal area using optimal path then the model is valid. With my model the robot takes 7steps to

reach the goal using optimal path ,this means that the model is valid. The visualization is below as shown with many dead ends.

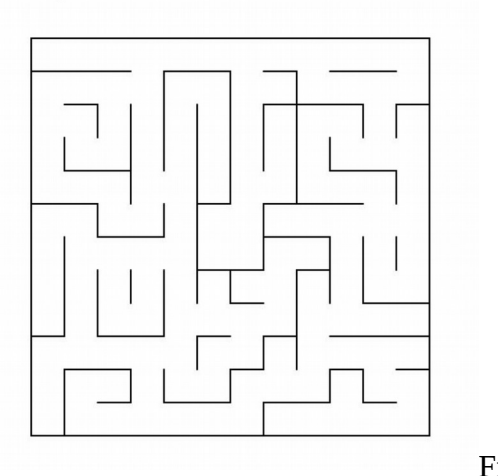

Fig 2 .A free form visualization of 12x12 maze

Reflection

The entire process of solving the maze problem can be summarized as such:

- Use the sensors to obtain or gain information about the maze.

- Combine this information with the heading and location of the robot.

- Update the maze map with any new information inferred directly or indirectly from the

sensor.

- Given the current knowledge of the maze, decide where to travel to maximize the

acquisition of new information.

- Decide how to get from the current location to the desired location(goal).

- Move toward the desired location.

- Repeat steps 1-6 until the optimal path from the start to goal locations has been found.

- Start the next run of the maze.

- Follow the optimal path from the start to the goal.

The interesting here is the fact that using this work flow in the implementation lead to solving the maze problem,the robot could be seen navigating from the start position and get to the goal using the optimal path in its second run which is exciting getting the problem solved. Also there are still some short

coming where I could not modify the test file to use in adding dead end,also others such checking if the robot is stock in a cell or loop,if an optimal path is found has been taken care of by the algorithm. This algorithm validated every set of data given for the project in which its achieved all the runs with required minimum steps using the optimal path and the reward which is the score in which am satisfied beyond every doubt my expectation has been made its is therefore recommended for solving such problems or related problems.

Improvement

The improvement to be made on the algorithm are:

- Higher speed

-Using the magnetometer for better turning control.

-Optimization of the small step forward at every dead-end found.

-Ability to return from the finish to the start automatically.

-Exploring the full maze to find the shortest path.

-Using dynamic speed to get higher speed: running faster on the long straight lines, while slowing down near the intersections.

There are other algorithm such as A* Algorithm,Breadth first search and Depth first search, Random turn and follow wall . Their use could also lead to a more efficient results since I did know how to implement it but will be an added advantage if I had a knowledge. From the solution I strongly belief there is still a model or an improved one that can get the solution better.

References

https://mcuoneclipse.com/2013/03/20/freedom-robot-solves-the-maze/

https://en.wikipedia.org/wiki/Maze_solving_algorithm

http://www.robotc.net/blog/category/stem/