Machine Learning Nanodegree

Capstone Proposal

15 April 2017

Domain background.

This project aims at programing a virtual maze to examine the performance and efficiency of a robot mouse to find it path from the corner to the center and navigate its path from start to the end. On each maze, the robot must complete two run. In the first attempt of the mouse to navigate the maze its tries to map the out the maze path to the center and find the best path to traverse from corner to the center, this first attempt takes more time, this gives it the privilege to explore all the route in the maze and its takes many steps to reach the center and in subsequent runs the robot mouse attempt to reach the center in the fastest time possible using the previously learned path. 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. 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.

Robotic motion planning has many applications such as used in the animation of realistic digital actors, simulation of ligand-protein binding and protein folding, and planning of complicated surgeries. Also a problem in this domain can be solved by simulating the world of a robot in its environment making decision on the most efficient means of achieving a goal.

Related research include Vladimir J.Lumelsky :A Comparative Study on the Path Length Performance of Maze-Searching and Robot Motion Planning Algorithms. In which he used the graph theory to find optimal path for the robot mouse which is one of the solution used in this proposal. Another related research is Vera Bazhenova 2010 on Robot Motion Planning ,he used A* algorithm and find the shortest path through the graph search algorithm.

F. Y. Annaz/ IJECCT 2012, Vol. 2 (2): A Mobile Robot Solving a Virtual Maze Environment in his book he did write on the use of virtual maze in examining the performance and efficiency of various types of robots with various on board algorithms to find the optimal path which is the main problem I will have to solve in this proposal.

Robotic motion planning brings about creativity, technical reasoning in solving problems and help in schools too to open the brains of student so that they will think out of the box even when solving problems.

Problem statement.

The main focus here is we want the robot to move from one end of the maze to the center of the maze that is movement is from the left corner of the maze to the center of the maze,taking the minimum steps possible,learn the environment fast and efficient at making decisions to the shortest path that leads quickest to the goal(center of the maze).This is done by decision making and training speed of the robot mouse on its own such its has to know its initial position and to know when its reaches the goal. In a series of test mazes and on each maze the robot complete two runs. The first run,staring from the initial point the maze is allow to explore all the path so as to know and master well such that in the second run the maze just go through shortest path without any interruption and faster using the minimum 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 and a maximum of one thousand time steps are allotted to complete both runs for a single maze.. This is obtain from its performance metric written mathematically as

Score = Nsr + 1/30Nfr

Nsr + Nfr <= 1000

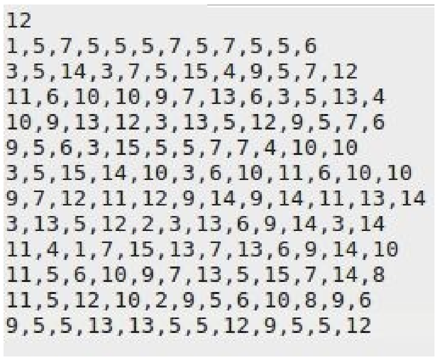
Nsr = number of steps for the second run,

Nfr = number of steps for first run.

Here, maze of any size can use this formula in other to get the score.

Dataset and input

Each maze has a specification and exist on an nxn of squares,n being even with a minimum n of 12 and max n of 16.Along the outside perimeter of the grid, and on the edges connecting some of the internal squares, are walls that block all movement. The robot will start in the square in the bottom- left corner of the grid, facing upwards. The starting square will always have a wall on its right side (in addition to the outside walls on the left and bottom) and an opening on its top side. In the center of the grid is the goal room consisting of a 2 x 2 square; the robot must make it here from its starting square in order to register a successful run of the maze. Mazes are provided to the system via text file. For n = 12 this is the specification data to produce a maze



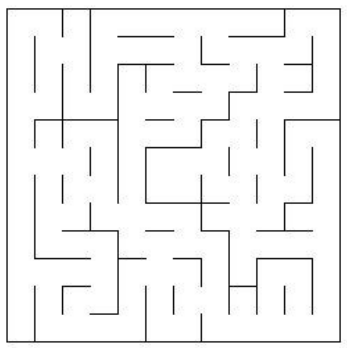
```
12
1,5,7,5,5,5,7,5,7,5,5,6
3,5,14,3,7,5,15,4,9,5,7,12
11,6,10,10,9,7,13,6,3,5,13,4
10,9,13,12,3,13,5,12,9,5,7,6
9,5,6,3,15,5,5,7,7,4,10,10
3,5,15,14,10,3,6,10,11,6,10,10
9,7,12,11,12,9,14,9,14,11,13,14
3,13,5,12,2,3,13,6,9,14,3,14
11,4,1,7,15,13,7,13,6,9,14,10
11,5,6,10,9,7,13,5,15,7,14,8
11,5,12,10,2,9,5,6,10,8,9,6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

Text file specification for the first maze.

On the first line of the text file is a number describing the number of squares on each dimension of the maze n. On the following n lines, there will be n comma-delimited numbers describing which edges of the square are open to movement. 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). Note that, 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.

For the robot to accomplish its task on this maze, its has sensors left,right and front sensors for obstacle detection whether the square is closed or open to ensure a successful run. on each 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. 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. It is assumed that the robot’s turning and movement is perfect. 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.

The maze visualization follows by writing ‘python showmaze.py test_maze_01.txt’ on the command line to view how a maze is produced by using information of maze specification from test_maze_##.txt,here n is equal # = 1,n = 12 and n/# can vary depending on the choice of the individual. The script uses the turtle module to visualize the maze,its read the maze specification carefully producing the maze by splitting each line as commas and build the required cell specification of the maze.



A visualized maze produced from ‘showmaze.py’

For this maze 01 chosen the robot is expected to do two runs first run its explores the cells as earlier mention and the second run is uses the best path for optimal path maximization and using 17 steps to reach the goal. Hence each cell in the maze visualization is a number of step that the robot will take to move.

Solution statement

We want to find the shortest path from start to finish. Two primary problems of path finding are (1) to find a path between two nodes in a graph; and (2) the shortest path problem—to find the optimal shortest path.   There are several algorithms to find shortest paths .Basic algorithms such as breadth-first and depth-first search address the first problem by exhausting all possibilities; starting from the given node, they iterate over all potential paths until they reach the destination node.

Graph traversal.

also known as **graph search**) refers to the process of visiting (checking and/or updating) each vertex in a graph. Such traversals are classified by the order in which the vertices are visited. There are two types of graph traversal that is depth-first search (DFS) and breadth-first search (BFS)

Breadth-first search (**BFS**) is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root or some arbitrary node of a graph, sometimes referred to as a 'search key' and explores the neighbor nodes first, before moving to the next level neighbors ,its expand all neighboring nodes progrssively through the graph until the target is is discovered. Hence its used to finding the shortest path between two vertices.

Depth-first search (**DFS**) is an algorithm for traversing or searching tree or graph data structures. One starts at the root selecting some arbitrary node as the root in the case of a graph and explores as far as possible along each branch before backtracking.

Dead-end filling algorithm : Dead-end filling is an algorithm for solving mazes that fills all dead ends, leaving only the correct ways unfilled. It can be used for solving mazes on paper or with a computer program, but it is not useful to a person inside an unknown maze since this method looks at the entire maze at once. The method is to 1) find all of the dead-ends in the maze, and then 2) "fill in" the path from each dead-end until the first junction is met. Note that some passages won't become parts of dead end passages until other dead ends are removed first. Therefore with all the end filled and only the solution left this offers a shorter path from the initial position to the target.

A* search algorithm :its is used for path finding and graph traversal. When given a starting position and a target position ,A* will progressively search through all the possible path of length increasing through the search space and around any obstacle until the target is found. As soon as the target is found that path of length becomes the the shortest path to the target. 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. It is formulated in terms of weighted graphs: starting from a specific node of a graph, it constructs a tree of paths starting from that node, expanding paths one step at a time, until one of its paths ends at the predetermined goal node .At each iteration of its main loop, A* needs to determine which of its partial paths to expand into one or more longer paths. It does so based on an estimate of the cost (total weight) still to go to the goal node. Specifically, A* selects the path that minimizes

$$f(n) = g(n) + h(n)$$

where *n* is the last node on the path, *g*(*n*) is the cost of the path from the start node to *n*, and *h*(*n*) is a heuristic that estimates the cost of the cheapest path from *n* to the goal. The heuristic is problem-specific. For the algorithm to find the actual shortest path, the heuristic function must be admissible, meaning that it never overestimates the actual cost to get to the nearest goal node.

Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph The algorithm exists in many variants; Dijkstra's original variant found the shortest path between two nodes, but a more common variant fixes a single node as the "source" node and finds shortest paths from the source to all other nodes in the graph, producing a shortest-path tree.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.

From the above algorithms and techniques its can be to deduced that all the techniques is been used by the robot mouse to learn and know its environment,learn the path so as to minimize the number of steps required to make an efficient move and use shortest time possible to get to the target (goal) through an optimal path in other to get its maximum score end not be penalized.

Benchmark Model

The robot has three obstacle sensors, mounted on the front of the robot, its right side, and its left side to help its detect the presence or absence of an obstacle so that its can take its move. Also for each maze the 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.

Evaluation metrics.

Solving maze involves two runs that is on each maze, the robot must complete two runs. In the first run, the robot is allowed to freely move round 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 and take shorter steps. 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. Therefore its is from this score that we tell the winner for the maze competition and the performance metric used here will be the score, its can be represented as;

Score = Nsr + 1/30Nfr

Nsr +Nfr <= 1000

range[0,1000]

Nsr = number of steps for the second run,

Nfr = number of steps for first run.

The score is used as performance metrics because only the score can tell if the maze made use of the optimal path using the shortest time possible and its the same score that differentiate between the winner of the competition. This score is calculated by taking the number of time steps required to execute the second run and adding it to one thirtieth the number of time steps required to execute the first run. This score formula is a linear function with a maximum score of 1000 and minimum score of 0 as such [0,1000] as the range. This applies for any test maze in the competition .

Project Design

A theoretical work flow for the proposal will be as follows:

-The robot need to know its initial position.

-learn the environment size and shape of the maze to explore its path.

-Determine the distance it can run and the direction to maximize the knowledge of the maze and minimize the number of steps during exploration.

-Its should have a knowledge of the goal and must get there

-Once the goal is reached check if the path is optimal

-Once the goal is reached save it and go for second run.

-Explore the maze using full knowledge obtain.

The robot need to know its initial position.

The robot will start in the square in the bottom- left corner of the grid, facing upwards. The starting square will always have a wall on its right side (in addition to the outside walls on the left and bottom) and an opening on its top side. This therefore means for the robot to not its initial position where its suppose to take a move the presence of the sensors precisely the will help to detect that there is a first opening to the fore front and obstacle on the sides so that this information is stored and the robot is aware of initial position such that its doesn't missed it and get to some sides.

learn the environment size and shape of the maze to explore its path.

For the robot to learn the environment its a full knowledge of the maze the shape which is a square and in two dimensions the first being the edge boundary bordering each cell in the maze which could be open or closed. This will help the robot know the walls and openings that are found in the maze. Also a knowledge of the shape of each cell and shape of the goal that its a four edge wall in the interior of the maze and the exterior of the maze is also a wall. This will enable the robot to be able to save it information and update it at any point in its run since knowledge on this will help to differentiate between openings edges,close edges ,open cells and closed cells and the shape and orientation of the goal. Therefore its will explore its path better.

Determine the distance it can run and the direction to maximize the knowledge of the maze and minimize the number of steps during exploration

This is to determine how far the robot can move in the maze having knowledge of the maze and can keep track of the edges therefore its time to explore every cell that the robot is not aware of finding its path through it and while doing this also trying to reduce the number of steps that the robot takes to get to the target and not so that its clear to the robot that its not about just through the maze but having to know that it has to get somewhere which it the goal an how far its goes is not the interest but how its gets to explore the cells to reach this target in the shortest path is what we are interested in also the robot needs to know the content of its four surrounding cells, that is, the content of the cell to its left, right, and ahead, in addition to prior knowledge of the previously revealed or stepped on cell (now the rear-cell). Therefore, during navigation, the robot sides vary according to the direction it is facing, hence it is important that the robot continuously updates the virtual environment of its direction so that the correct new surroundings are revealed to it.

Explore a path to the goal and get to the goal.

To to the gaol by the robot,knowing the the shape of the goal area as a 2x2 square of cells with a single opening from the right corner interior of the maze,the robot can have this knowledge to get to the target by finding single opening or noticing seven wall of the target. once the entrance to the goal is discovered the robot can continue to explore other path. The A*search algorithm can be used to find path since its uses **heuristic technique**(meaning "find" or "discover") is any approach to problem solving, learning, or discovery that employs a practical method not guaranteed to be optimal or perfect, but sufficient for the immediate goals,heuristic methods can be used to speed up the process of finding a satisfactory solution and can be a mental shortcuts that ease the cognitive load of making a decision. Hence can be used to find the path to the goal.

Once the goal is reached check if the path is optimal.

This simply means that the robot has found the best path through the maze from its initial position bottom left to the goal center of the maze using the wall map its has drawn to learn the maze. Also its means every edge and every cell opening in the maze is well known and this algorithm **Dijkstra's algorithm** as earlier discussed is used also to check if its optimal or shortest path to the goal. Therefore the target of reaching the goal in the quickest time possible and shortest path with reduced steps has been found called the optimal path.

Once the goal is reached save it and go for second run.

Having reached the goal be the robot,and explore other path,the robot has to return to the initial position using reset so as to go for the second run which it the exploitation run keeping track of all the knowledge of the movement and rotation its has previously used to get to the target during exploration run, but now taking and efficient move following all the instructions that pertain to an optimal path which is the shortest path,reduced steps and shortest time possible and get to the target.

As concerns data analysis for this proposal,there is no data preprocessing needed since the robot gets the data needed by interacting with the maze environment and sensing walls with its sensors, and once its has gotten sufficient data for its algorithm to work,its starts using it on the maze hence direct and no need for data preprocessing.