Machine Learning Engineer Nanodegree

Capstone Proposal

Shashank P
November 29th, 2017

Plot and Navigate a Virtual Maze

Domain Background

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 mice are completely autonomous robots that must find their way from a predetermined starting position to the central area of the maze unaided. The mouse will need to keep track of where it is, discover walls as it explores, map out the maze and detect when it has reached the goal. Having reached the goal, the mouse will typically perform additional searches of the maze until it has found an optimal route from the start to the center. Once the optimal route has been found, the mouse will run that route in the shortest possible time.

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

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.

The aim of this project is to balance the tradeoff between time taken for exploration or learning and time taken to reach the goal in the shortest time with the paths found during the exploration phase. The robot's score is a quantitative measure of this tradeoff.

Datasets and Inputs

The dataset is the virtual maze. The maze exists on an n x n grid of squares, n even. The minimum value of n is twelve, the maximum sixteen. 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.

Robot receives input from its sensors. 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.

Solution Statement

Dynamic Programming

Dynamic Programming is used to find the shortest path to the destination from any point in the maze. In dynamic programming, the distance from any point on the maze to the destination, called the 'distance value', is calculated and stored as the robot explores the maze. The distance value of all cells in the maze is updated on each move of the robot. During the exploration phase, the robot can be made to traverse the path of decreasing 'distance value' to make sure that it reaches the goal at least once. Once the goal is found, a policy for this path can be formulated. For each of the policies found, the time taken can be calculated and the one with the minimum time is chosen as the 'solution policy'. The exploration phase can be stopped after a certain number of steps or after the robot has explored certain percentage of the maze (say 80%). During the second run, the 'solution policy' is used to reach the goal.

Benchmark Model

Robot's score = number of time steps required to execute the second run + (number of time steps required to execute the first run)/30

Maze 1 (test_maze_01.txt) :

Number of time steps required to execute the first run = 300
Number of time steps required to execute the second run = 23
Benchmark score = 33

Maze 2 (test_maze_02.txt) :

Number of time steps required to execute the first run = 450
Number of time steps required to execute the second run = 32
Benchmark score = 47

Maze 3 (test_maze_03.txt) :

Number of time steps required to execute the first run = 600
Number of time steps required to execute the second run = 37
Benchmark score = 57

Evaluation Metrics

The evaluation metric for this project is the robot's score which is

Robot's score = number of time steps required to execute the second run + (number of time steps required to execute the first run)/30

Robot's score is a tradeoff between the time taken for exploring and time taken by the optimum solution found. During the first run, the robot explores the maze and the number of steps taken during this run is the time steps taken for exploration. During the second run, the robot reaches the goal using the optimum policy found. The time steps taken by this optimum policy plus one thirtieth the number of time steps required to execute the first run makes the robot's score. Lesser the robot's score(on an average), better is its performance.

Project Design

The algorithm that I will be using is dynamic programming.

First run:

Initially the 'distance values' of each cell in the maze is initialised assuming that there are no walls in between, with the 'distance value' of goal being 0 and increasing thereafter. The robot starts from the location [0, 0]. The robot moves to the next cell location which has lesser 'distance value' than the present cell. If there is more than one choice, it chooses the next cell randomly among the possible choices. The intention of making the robot go towards lower 'distance value' is to make the robot reach the goal. Once the robot reaches the goal, the policy is calculated. Now, to make the robot move back to the start, the 'distance values' are changed so as to make goal, the source, and start, the destination, with 'distance value' of 0. Now the robot tries to reach the start location and policy is calculated. After each new policy is found, it is compared with the previous optimum policy and the optimum policy is changed as required. This process is repeated for a certain number of steps or until the robot has explored certain percentage of the maze(say 80%).

Second run:

The optimum policy found during the exploration phase is used by the robot to reach the destination.

References

1) Udacity course on Artificial Intelligence for Robotics (cs373)
2) Flood fill algorithm by Srikanth Pagadala https://www.youtube.com/watch?v=NBSps5kW_qg