Machine Learning Engineer Nanodegree

Capstone Proposal

Hai Xiao

January 9th, 2017

Proposal

Plot and Navigate a Robotic Agent in a Virtual Maze

Domain Background

Robot Motion Planning is a typical AI problem to autonomous agents, according to wikipedia it is also known as the navigation problem in robotics for the process of breaking down a relocation or reconfiguration task into discrete motions that satisfy environment constraints and possibly optimize some aspect of the movement. It ranges from a pure robotic control (e.g. robotic arm) to both learning and control (e.g. exploration and searching, SLAM), in both known and unknown environments, continuous and discrete space, or deterministic and stochastic controls.

This project takes inspiration from Micromouse competitions (checkout 2014 Micromouse competition Youtube video). The maze solving problem is one of the most popular ones in the field of robotics. In fact IEEE designed a Micromouse competition dating back to the 70s, it gains popularity all over the world, and the problem itself has become one of the most accessible challenges in robotics teaching, experiment and even research. Different than to build a real robotic mouse navigating a real maze as people do in the competition, our task in this project is to come up with an efficient policy or model for a virtual intelligent agent (robotic mouse) to explore, discover and plot (follow determined) paths from a corner of the maze to its target at the maze center.

Problem Statement

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, simplified model of the world is provided along with specifications for the maze and robot, but environment should not be considered all known, for example no presumed maze configuration (even within its basic specification) or size should be made (but 3 different testing mazes of size 12x12, 14x14 and 16x16 must be validated); project objective is to obtain generalized methods in achieving the fastest time possible to a series of test mazes. The problem should not be considered a pure planning problem, provided that environment is not 100% known, at minimum some sort of learning and/or exploration should be considered in the solution (particularly this applies to the 1st run of the robot in maze). The score of a solution will be weighted sum of the average learning time steps and traverse time steps (post-learn or search).

Datasets and Inputs

Solution will not be targeted to specific mazes, and 3 pre-configured mazes are provided as test set, mostly used for benchmark comparison, discussion and reporting.

It is suggested that we come up with our own maze using test mazes as templates. All 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.

When solution is not built upon supervised learning, we are not required to strictly follow train/validation/test dataset split scheme, as long as the solution be a generalized design, which reflects the robustness to achieve the goal and perform on test mazes. Otherwise potential issue or limitations of the approach should be provided with discussion of the maze as well. There is no sensitivity of the data due to the Free-Form nature of this project, both methods and maze configuration.

Solution Statement

Due to non-fixed environment, robot should not assume the maze map before start, except for very minimum info pre-defined, such as traverse rule (including its action tracking as odometer, turn), location of start (left-bottom corner), location of target (maze center). So viable solution should be based on some level of learning capability vs. a pure search over a preloaded graph, a search algorithm that combines both traverse (search) and localization (know surrounding with sensors reading) capabilities may be viable as well.

To address this problem, two solutions are provided:

- Q-Learning based solution
    - Particular suited for initial map (or partial map) exploration
    - Optimal route (policy) is implicitly resolved at end of the learning
    - Challenge: optimize the learning phase, so the target can be reached at minimum steps in average
- A* search based solution
    - harness the sensors L/F/R view to construct reachable local connectivity at any given point of time
    - No rear view sensor, this seems to be a drawback initially, but if we eliminate the backup move, then 3-side sensors read is ok in discovering and exploring neighbors at each time step (N.B. this improvement freed search on pre-loaded graph)
    - Use explored cell in maze to track the distance from START (left-bottom corner)
    - Use reachable cell in maze to track the heuristic distance to TARGET (assume the maze size is given)
    - Track robot current coordinate from all prior movement (so agent roughly knows its relative location to TARGET, this basic location tracking is KEY to all localization agent, in the future we will see advanced version of this using particle filter in SLAM)
- Common challenges (model various [1-3] fwd steps of the robot)

    - Q-Learning: model them as different actions
    - A*: ignore it at exploration and search, convolute/compress at the end on path determined

Benchmark Model

Ideally there are TWO separate benchmarks in decomposition, one is learning/exploration time steps for the 1st run, second is the movement steps for the 2nd run, both are smaller the better.

To the benchmark of the 2nd run, we can use shortest path from A *search on pre-constructed full map (N.B. this may be different than A* search based solution above) as best benchmark.

The benchmark of the 1st run is little tricker, and it determines how well a learning or space searching algorithm behaves. It is not viable to look for real Micromouse benchmark as they (competitions) were all time base, which factored in more dynamics of the real system. To obtain a good reference for how the learning/exploring phase is doing, we can set up the left-hand-rule steps or simply 2*N*N (N is 12, 14, 16 respectively) as an acceptable upper bound (acceptable lowest benchmark).

Evaluation Metrics

We will use combined SCORE of first run and second run as final score to evaluate each experiment:

- If SCORE >= 1,000, then FAIL
- If SCORE < 1,000, then PASS

To minimize bias, we will use median SCORE of 100 runs (regardless FAIL or PASS) to evaluate the performance of a solution implementation.

The performance of second run seems to be heavier weighted than the 1st run (1 vs. 1/30), but it is also important to have qualified runs for both, i.e. total_steps <= 1,000

Project Design

In this final section, summarize a theoretical workflow for approaching a solution given the problem. Provide thorough discussion for what strategies you may consider employing, what analysis of the data might be required before being used, or which algorithms will be considered for your implementation. The workflow and discussion that you provide should align with the qualities of the previous sections. Additionally, you are encouraged to include small visualizations, pseudocode, or diagrams to aid in describing the project design, but it is not required. The discussion should clearly outline your intended workflow of the capstone project.

Data Pre-processing

- The maze specification and robot's sensors reading and movement (none-stochastic) is provided and 100% accurate, there is no data pre-processing or calibration necessary.

Implementation

Classes (Data Structure) for maze

Classes (Data Structure) for robot

Two Algorithms (as in solutions) , and workflow items

- Q-Learning
    - Need to set step length (limit) of each learning episode, e.g. <= 2x*N*xN
    - How to set different immediate rewards at different criteria? How to set initial Qs, agnostic to maze walls configuration?
    - Need to tune hyper parameters properly, e.g. Alpha (learning rate), Gamma (discount factor), and epsilon
    - Need to evaluate the size of Q table, if it is too big then need to use function approximation instead of table
- A* search
    - Justify the 3-side sensors reading vs. all 4-sides, and approximation built on top of it
    - Justify the variable fwd-step length (1-3), in this implementation
    - Justify the choice of heuristic function in estimate length to the target, and its pros vs. cons, particularly on very different maze configurations
    - What this falls back to if maze dimension and target location are not provided
- To both solutions
    - Consider how to avoid dead-end efficiently
    - Consider how to avoid loop route efficiently
    - Discuss what if robot's movement is non deterministic
    - Discuss what if robot's sensors reading is non deterministic
    - Discuss what if the space is continuous vs. discrete as in real competition
- Comparison, Visualization and Analysis
- Conclusion

N.B. Some details of this session is unfilled due to lack of details. But it outlines high level designs and reporting structure.