RL based Maze Solver

Open Projects 2021

PROJECT REPORT

TEAM MEMBERS

- Apurba Prasad Padhy (ECE)
- Chirag Arora (ECE)
- Rishabh Dubey (EE)
- Tushar Sahu (ECE)
- Yash Bhinwal (PSE)

MENTORS

- Agrim Agrawal(EE)
- Annu Shree (MT)
- Vansh Goyal(CE)

INDEX

ABSTRACT

The aim of this project is to solve a given 2D maze (given as an image input) using RL (Q-Learning) algorithm. The program first processes the image input of the maze and converts it into a matrix. The Q-Learning algorithm then uses this matrix to generate a q-matrix, which is finally used to get the shortest path. The program can also generate random mazes and then solve it.

MOTIVATION

Reinforcement Learning in robotics has been a challenging domain in the field of AI for the past few years. The ability to equip a robot with a powerful enough tool to allow an autonomous discovery of an optimal behavior through trial-and-error interactions with its environment has paved the path to many deep research projects.

Our project inspires us to solve many problems, like Autonomous Mobile Robot Obstacle Prevention using Q-learning.

SOFTWARE ASPECT OF THE DESIGN

- Python - It is an interpreted high-level object-oriented programming language designed by Guido van Rossum. It emphasizes code readability with significant use of indentation. It has tons of third-party open-source libraries (which can be installed using its own package manager *pip*) to assist all types of programs.
- Jupyter Notebook - It is a product developed under *Project Jupyter* (spun off from *IPython* in 2014 by Fernando Perez) and is used to create documents containing live code, equations, visualizations and narrative text. It is used in data cleaning and transformation, numerical simulation, statistical modeling, data visualization and machine learning.

SOFTWARE ASPECT OF THE DESIGN

- NumPy - It is a Python library providing fundamental package for scientific computing. It is used for working with arrays in the domain of linear algebra, fourier transform, matrices, etc. Numpy arrays are 50x faster than Python lists.
- Matplotlib - It is a Python library for creating static, animated, and interactive visualizations and plots and was introduced by John D. Hunter in 2002.
- Python Imaging Library (PIL) - It is a Python library providing support for opening, manipulating, and saving many different image file formats. It supports Python version 1.5.2 to 2.7. A subsequent fork of the PIL repository named *Pillow* added Python 3.x support.

REINFORCEMENT LEARNING

Reinforcement Learning is a type in machine learning that comes under Artificial General Intelligence (AGI). This type of learning does not require any inputs\outputs or any intermediate actions to achieve a task. Instead, it finds a trade-off between exploration and exploitation. This make use of an idea of training an agent in an unknown environment, which has an analogy to make a child learn from basic.

Q- Learning is a model free, off-policy reinforcement learning method that investigates the environment based the quality of actions take in each state. It is an off-policy learning because it even uses actions outside of the current policy to explore the environment. 'Q' stands for quality of actions.

APPLICATIONS

This project can be used to develop a navigation system with the ability to learn to adapt to unknown environments. Such game-playing AIs are designed in a way that their solutions are relevant in many practical applications :

- Industrial Applications - Mobile robots have been increasingly used over the last two decades in various industries to move goods from one place to another with optimal movement policy.
- Restaurants of the Future (ROTF) - ROTF is one of the greatest physical application of the project. Since such a RL based implementation in a mobile robot will help us to give and take orders from the customers, in a faster and more efficient way.
- Navigation - Maze solving can further be extended for autonomous navigation in an occupancy grid to get to the nearest landmark like an EV charging station or a petrol pump.

ACCOMPLISHMENTS

- Defining the Agent of the ML Algorithm using its various positions and movements in the maze
- Setting up the definition and functionality of Maze with high degree of visualization and scoring:
    - Function for checking correct positions and calculating possible moves
    - Function for checking if the Agent won
    - Function for scoring any move of the Agent
    - Function for visualizing the maze and positions of Agent using Matplotlib

ACCOMPLISHMENTS

- Setting up Q-Learning using Q-Matrix and mapping matrix indices to movements of the agent
- Setting up function to generate random mazes with start and end as top-left and bottom-right corners of the Maze
- Setting Explore and Exploit conditions and testing the above setup episode-wise
- Generalizing the random maze generate function to set up any arbitrary point as start and end coordinate
- Setting up a function to visualize the path taken by the Agent to traverse the Maze using Matplotlib

ACCOMPLISHMENTS

- Code for preprocessing of the input maze image using PIL:
    - Reading the input image and converting the black pixels to 1 and white to 0.
    - Function for checking identical consecutive rows and columns in the matrix and then deleting them to reduce the matrix size
    - Function for inverting the reduced matrix, trimming the extra whitespace and detecting the openings of the Maze
    - Function to remove undesired pixels that appear at the corners of walls, that block the paths and that appear due to round edges of the walls

LIMITATIONS

At the current state, our project has a few limitations :

- Before running, the program requires total number of episodes and number of random episodes to be given as a manual input. These can be developed as a function of the maze size to get a higher degree of automation.
- The reduceMatrix(), trim() and sharpen() functions also require manual passing of parameters.
- The Image Processing part of our project is not strong enough to process very low-resolution maze images.
- The simple reinforcement learning algorithm would collapse when dealing with complex mazes.

FURTHER IMPROVEMENTS

In future, we plan to :

- Develop total number of episodes as well as number of random episodes as a function of maze size for more optimal functioning of our program i.e., F(s) = N(E).
- Generate an animated maze solution.
- Implement Multi-Objective target search, wherein the agent of Q-Learning must visit intermediate flag positions before going to the end of the maze.
- Improve the Image Processing part of our project to an extent where we can process very low-resolution images and get automated inputs to the reduceMatrix(), trim() and sharpen() functions.
- Extend our project to 3D Mazes
- Integrate a broader aspect of deep learning.

REFERENCES

- https://www.youtube.com/watch?v=rfscVS0vtbw
- https://www.programiz.com/dsa/graph
- https://www.programiz.com/dsa/dijkstra-algorithm
- https://www.programiz.com/dsa/bellman-ford-algorithm
- https://www.samyzaf.com/ML/rl/qmaze.html
- https://www.youtube.com/watch?v=qhRNvCVVJaA
- https://en.m.wikipedia.org/wiki/Q-learning
- https://stackoverflow.com/questions/57610416/how-to-read-a-maze-from-an-image-and-convert-it-to-binary-values-in-python