A Project Report on:

Maze Solving using Deep Reinforcement Learning

Submitted in partial fulfillment of requirment for the award of degree of:
Bachelor of Engineering
in
Computer Engineering

by
L.Aniruth Naraayanan(16102005)
Siddesh Kokane(16102054)

Under the Guidance of:
Jaya D Gupta

Department of Computer Engineering
A.P. Shah Institute of Technology
G.B.Road, Kasarvadavli, Thane(W), Mumbai-400615
UNIVERSITY OF MUMBAI
Academic Year 2017-2018

Approval Sheet

This Project Report entitled “**Maze Solving using Deep Reinforcement Learning**” Submitted by **“L. Aniruth Naraayanan”(16102005)** and **“Siddesh”(16102054)** is approved for the partial fulfillment of the requirement for the award of the degree of **Bachelor of Engineering** in **Computer Engineering** from University of Mumbai.

Prof. Sachin Malawe
(HOD Computer Engineering)

Jaya D. Gupta
(Project Guide)

Place: A. P. Shah Institute of Technology, Thane
Date :

Certificate

This is to certify that the project entitled "**Maze Solving using Deep Reinforcement Learning**" submitted by **"L. Aniruth Naraayanan"(16102005)** and **Siddesh Kokane (16102054)** for the partial fulfillment of the requirement for award of a degree **Bachelor of Engineering** in **Computer Engineering** tothe University of Mumbai,is a bonafide work carried out during academic year 2017-2018.

Jaya D Gupta
Guide

Prof. Sachin Malawe HOD Computer Engineering	Dr. Uttam D. Kolekar Principal

External Examiner(s):

1.

2.

Place: A. P. Shah Institute of Technology, Thane
Date :

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

L. Aniruth Naraayanan"(16102005)

Siddesh Kokane (16102054)

Date:

Contents

1. Abstract

In this project we aim to create a solution to solve any MXN maze using machine learning techniques. Specifically we are using Deep Reinforcement Learning , a combination of Deep Neural Networks and Reinforced Learning .

The input maze will be a randomly generated array containgn information such as free cell, blocked cell, target etc., which will be fed to a tkinter window and plotted using matplotlib. This project willl run on a Tenserflow back-end with keras being used as the neural network library.

This project can be useful in real-world application such as path finding, mapping applications, robotics, etc..

2. Introduction

2.1. Maze Solving:

Traditional maze puzzles have been used a lot in data structures and algorithms research and education. The well-known Dijkstra shortest path algorithm is still the most practical method for solving such puzzles, but due to their familiarity and intuititive nature, these puzzles are quite good for demonstrating and testing Reinforcement Learning techniques.

2.2. Reinforcement Learning:

Reinforcement learning is a machine learning technique for solving problems by a feedback system (rewards and penalties) applied on an agent which operates in an environment and needs to move through a series of states in order to reach a pre-defined final state. A classical example is a rat (agent) which is trying to find the shortest route from a starting cell to a target cheese cell in a maze (environment). The agent is experimenting and exploiting past experiences (episodes) in order to achieve its goal.
It may fail again and again, but hopefully, after lots of trial and error (rewards and penalties) it will arrive to the solution of the problem. The solution will be reached if the agent finds the optimal sequence of states in which the accumulated sum of rewards is maximal (in short, we lure the agent to accumulate a maximal reward, and while doing so, he actually solves our problem). Note that it may happen that in order to reach the goal, the agent will have to endure many penalties (negative rewards) on its way. For example, the rat in the above maze gets a small penalty for every legal move. The reason for that is that we want it to get to the target cell in the shortest possible path.
However, the shortest path to the target cheese cell is sometimes long and winding, and our agent (the rat) may have to endure many penalties until he gets to the "cheese" (sometimes called "delayed reward").

3. Maze Generation

The following presents the algorithm behind the generation of a random m x n maze.

Input :

m – no. of rows
n – no. of column

Procedure:

1. Create a m x n maze with all cells blocked – (0 in numpy array)
2. Randomly traverse the array
3. Mark every visited block as a free cell – (1 in numpy array)
4. If current cell is target cell then exit
5. Repeat IE. go to 1

Output:

Returns a numpy array containing information about all free cells, blocked , agent and target cell.

3. Maze Solving

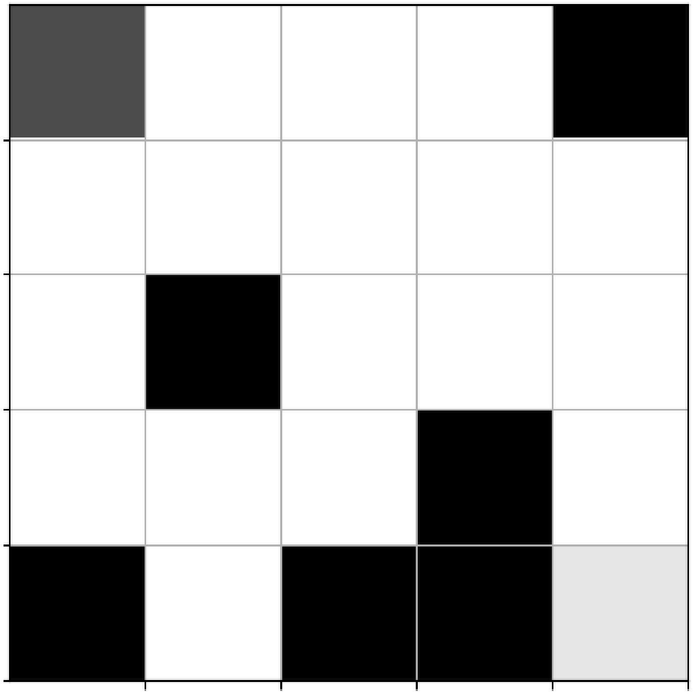
The following procedures explain the how the generated maze is solved using reinforcement learning and neural networks.

3.1. Step 1 : Feed the maze to the program from the generated input array

Input :
m x n numpy array, which holds all the cells of the array and also denotes if a cell is free or not.
By default the Agent starts from top-left (0,0) and Cheese is located at bottom right(n-1,m-1) .

Procedure :
1. Initialize maze properties using Qmaze class
2. Set functionalities such as valid actions, reward, min reward, state, etc.. to define agents behavior after every iteration.
3. Plot the input Maze on to the tkinter window using matplotlib.

Sample input maze:



3.2. Step 2. Build a neural network model and train it

1. The most suitable activation function is **SReLU (the S-shaped relu)**
2. Our optimizer is **RMSProp**
3. Our loss function is **mse (Mean Squared Error)**.

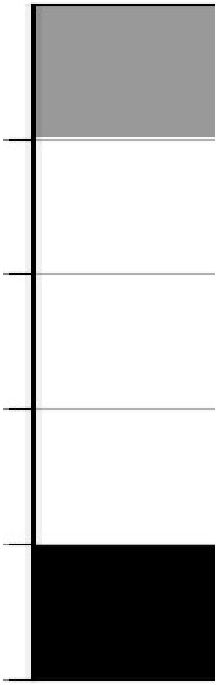
This model is saved in a json file saved and used for solving any number of mazes

Step 3. 3. Solve a maze using the model

The above generated model or a previously used model can be used to train and solve a maze.

Finally plot the maze on the tkinter window to display the output

Solved maze:



5. Technology Stack.

Programming Languages	**–**	**Python3 (v 3.5.2)**
Machine Learning Back-end	**–**	**Tensorflow**
Neural Network Library	**–**	**Keras (v 2.2.4)**
Operating System	**–**	**Ubuntu(Linux) (v 18.04 LTS)**

6. Conclusion and Scope

Thus, we have implemented an machine learning algorithm to solve a randomly generated maze.

This may not be the most efficient method of solving mazes as classical algorithm such as Dijkstra's algorithm, A star algorithm etc .. as much more efficient, but taking part in this project has given us a firm understanding of reinforcement learning and how its implementation .

This approach can be used in dynamic circumstances such as a maze with the agents and the target moving simultaneously in real time etc. and can even beat classical approaches when it comes to efficiency. Thus this project does have scope in other circumstances and cases.

Acknowledgement

We would like to thank our guide Ms. Jaya D. Gupta for guiding us throughout this project and our institute APSIT for providing us this opportunity to perform this project

References

1. Demystifying Deep Reinforcement Learning (https://www.nervanasys.com/demystifying-deep-reinforcement-learning)
2. Keras plays catch, a single file Reinforcement Learning example (http://edersantana.github.io/articles/keras_rl)
3. Q-learning (https://en.wikipedia.org/wiki/Q-learning)
4. Keras plays catch - code example (https://gist.github.com/EderSantana/c7222daa328f0e885093)
5. Reinforcement learning using chaotic exploration in maze world (http://ieeexplore.ieee.org/document/1491636)
6. A Reinforcement Learning Approach Involving a Shortest Path Finding Algorithm (http://incorl.hanyang.ac.kr/xe/paper/ic/ic2003-3.pdf)
7. Neural Combinatorial Optimization with Reinforcement Learning (https://arxiv.org/abs/1611.09940)
8. Google Acquires Artificial Intelligence Startup DeepMind For More Than $500M (https://techcrunch.com/2014/01/26/google-deepmind)
9. How DeepMind’s Memory Trick Helps AI Learn Faster (https://www.technologyreview.com/s/603868/how-deepminds-memory-trick-helps-ai-learn-faster/?imm_mid=0ef03f&cmp=em-data-na-na-newsltr_20170320)
10. Methods and apparatus for reinforcement learning US 20150100530 A1 (Patent Registration) (https://www.google.com/patents/US20150100530)
11. Introduction to Reinforcement Learning (http://www.cs.indiana.edu/~gasser/Salsa/rl.html)
12. Reward function and initial values: Better choices for accelerated Goal-directed reinforcement learning (https://hal.archives-ouvertes.fr/hal-00331752/document)
13. Andrej Karpathi blog: Deep Reinforcement Learning: Pong from Pixels (http://karpathy.github.io/2016/05/31/rl/)