Designing Reinforcement Learning Model For Decision-Making & Implementing Graph Theory For Shortest Path

Reena Hooda
Assistant Professor, Dept of CSE
Indira Gandhi University Meerpur (Rewari)- Haryana -122502,
India.

Abstract- *Reinforcement leaning as a branch of artificial intelligence is a target oriented learning method that based on the negative and positive rewards in a stream of actions and set of states thereof. It is not exactly the machine learning or deep learning, it's like a bot and semi supervised methodology. The paper pondered the importance of the reinforcement learning paradigm in enhancing the computing powers of deep learning. Further, the paper implement the graph theory and Dijkstra bidirectional graph containing rewards and discount factors on nodes connected with edges in order to computer shortest path. The paper also tags the Q theory to count the probabilistic distribution of rewards to select the optimal path.*

Methodology used: *The graphs are generated in JupyterNotebook (anaconda3) and Spyder(Python 3.7). Rewards table are created in word.*

Keywords- Rewards, Penalties, Gamma, Discount Factor, Graph, Score.

I. INTRODUCTION

Reinforcement learning is known supervised learning method that works on sequence of rewards and penalties to reach the target within a time constraint. The next step in the sequence is totally based on current decision or action of the agent and its feedback system therefore involving the feedback & taking the next steps always involves a delay. [1] The basic difference between reinforcement learning and supervised method is that in supervised learning model for instance K nearest neighbor, method all the input values are given at initial position, for example value of key, Training data set and target are given at initial point; in reinforcement learning the agent is supplied with the current information and the target only. The input is not received at the single initial entry stage however; agent received the input in sequences after each action. Though the agent in advance knows its target,thus reinforcement learning is a semi-supervised learning method. [2] So the application areas also different for both of the techniques for example K nearest neighbor method is suitable to find the banking frauds etc. and reinforcement learning method is suitable for robotics and machine learning or the industry automation etc., take an example of game in reinforcement learning the model learn from a game, the model update itself to enhance its previous performance where the K nearest neighbor method is not a learning method, K nearest neighbor method takes input at a time initially and neverdo any learning. It takes the decision based on the K, sampling and the target at a time. Reinforcement learning retain its previous data and experiences so that it can improve its future actions. This way, under the same circumstances the model can act differently from the previous actions as it learns from the past experiences to take the current decision apart from the reward or penalties.[3]

II. GENERATING GRAPH AND DESIGN THE REWARDS TABLE

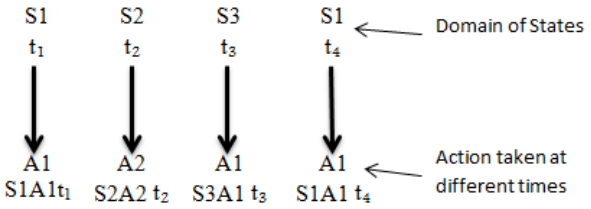
Apart from the environment, rewards, penalties, values and state, the few more consideration in reinforcement learning like discount factor and Q value can also plays an important role in decision making. [3]

Discount factor plays a very crucial role in minimizing the delay in traversing as the model works on immediate action to drive the agent to select certain parts only. By taking the future rewards in multiplication with the discount factor bound the agent to what to do and what not to do. [3]Q value includes the extra parameter to count the long-term rewardsof the action taken.As the motive of the model is to enhance the long term rewards therefore, it can even receive low immediate rewards to attain long-term high rewards by considering Q value, along with the immediate value of the states. [3]

Current status of an agent is sum of all the rewards it gained from its previous steps such as:

$$\sum_{t=0,count=0}^{t=\infty,count\leq t} R_i\left(S_{i,count}(t_j), A_i(t_j)\right);$$ Where:

t=Time
A=Actions associated with each State at time t
S= Set of States at time t.
R= Rewards associated with each Action.



Now, the main issue is the selection of the rewards whether it is fixed for each action or is it different based on the time. For simplicity, it is better to implement reinforcement learning initially at a fix number of possible state or in a limited sized domain like repetitive task performed in manufacturing of products or parts, heat checker, or industry automation etc. [4] discount factor gamma (y) can beincluded by associating with particular actions only; that is the multiple of the future state's rewards to increase the rewards immediately as given in figure 2. This can be seen as gaining more coins at a particular state or the penalties in a specific path. Negative positive rewards controls the behaviours of an agent, for instance, positive rewards increase the probability that some particular steps may be taken and repeated to increase the rewards quickly. Negative rewards can bound the agent by reflecting the possibilityofpunishment anddirect to avoid such passes. [5]

```
# -*- coding: utf-8 -*-
"""
Created on Sun Apr 18 14:03:21 2021

@author: anshi
"""

import numpy as np
import pylab as piGR
import matplotlib.pyplot as plott
import networkx as nxtN
Re_GraphBoxpoints = [(1, 2), (1, 6), (2, 3), (2, 7), (3, 4),
         (3, 8), (4, 5), (4, 9), (5, 10), (6, 7),
          (6, 11), (7, 8), (7, 10), (8, 19), (8, 13),
          (9, 14), (9, 10), (10, 15), (11, 12), (11, 16),
          (12, 17), (12, 13), (13, 18), (13, 14), (14, 15),
          (14, 19), (15, 20), (16, 17), (16, 21), (17, 22),(17, 18),
          (18, 19), (18, 23), (19, 24), (19, 20), (20, 25),(21, 26),
          (21, 22), (22, 23), (22, 27), (23, 24), (23, 28),(24, 29),
          (24, 25), (25, 30), (26, 27), (27, 28), (28, 29),(29, 30)]
target = 70
tar_graph = nxtN.Graph()
tar_graph.add_edges_from(Re_GraphBoxpoints)
locationG = nxtN.spring_layout(tar_graph) #nodes position
nxtN.draw_networkx_nodes(tar_graph,locationG,node_color="r", node_size= 500)
#nxtN.draw_networkx_nodes(tar_graph,node_color="r")
nxtN.draw_networkx_edges(tar_graph, locationG,edge_color="g")
nxtN.draw_networkx_labels(tar_graph, locationG)
#piGR.show()
plott.show()
```

Figure 1: Source code of the graph generated having 30 nodes in total.

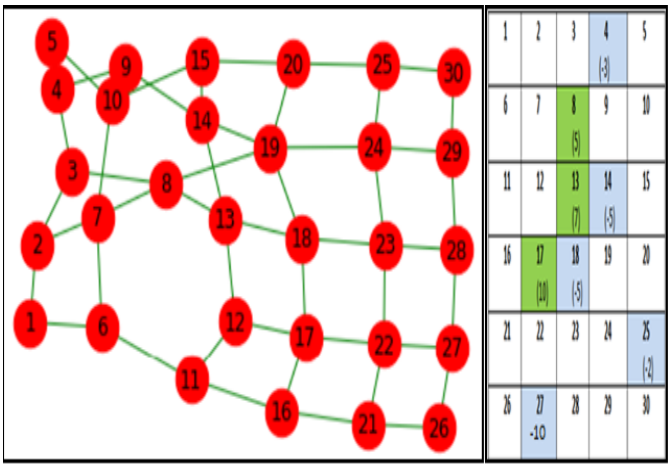


Figure 2: Graph generated from source code and table showing the gamma factors.

For instance, if node 1 is the source and node 30 is the destination; some random paths have been selected from 1 to 30 and calculated on the basis of table also known as rewards table [8], the gamaa or discount (y) factors [3] are 8,13, 17 positive rewards and 4,14,18,25,27 are negative rewards otherwise, 5 rewards for each non- discount nodes (left nodes) as shown in figure 3.

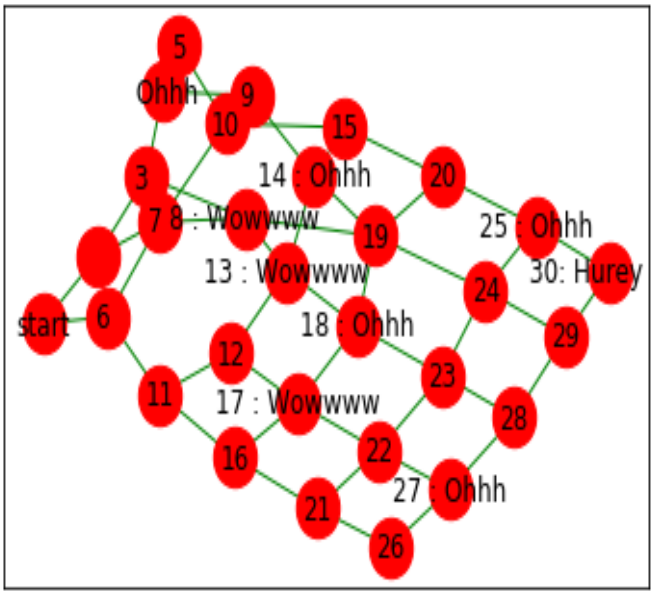


Figure 3: Shows the graph with rewards table data.

The figures are different due to randomly designed graph by network library, figure 3 shows the discount factor for positive rewards as "Wowwww" for nodes 8,13,17 and negative rewards as "Ohhh" for nodes 4,14,18,25,27 and source code is given in figure 4.

```
"""# to generate the undirected graph  [9] from rewads table in figure 2.
#Showing the Nodes name with rewards and discount factors,  along wiht source
and destination. [9]"""

Neg_Rewards = [4,14,18,25,27]
Pos_Rewards = [8, 13, 17]
tar_graph = nxtN.Graph()
tar_graph.add_edges_from(Re_GraphBoxpoints)
Rewards_Table = {1:'start', 2:'',3:'3 ', 4:' Ohhh',5:'5',6:'6 ',
                 7:'7 ',
           8:'8 : Wowwww', 9:'9 ',10:'10',11:'11',12:'12',
            13:'13 : Wowwww',14:'14 : Ohhh', 15:'15', 16:'16', 17:'17 : Wowwww', :
            19:'19',20:'20',
          21:'21',22:'22',23:'23',24:'24',25:'25 : Ohhh',26:'26', 27:'27 : Ohhh',
          28:'28',29:'29', 30:'30: Hurey got'}

New_graph = nxtN.relabel_nodes(tar_graph, Rewards_Table)
locationG = nxtN.spring_layout(New_graph)
nxtN.draw_networkx_nodes(New_graph, locationG, node_color="r", node_size= 500)
nxtN.draw_networkx_edges(New_graph, locationG, edge_color="g")
nxtN.draw_networkx_labels(New_graph, locationG)
siz = plott.figure() #[9]
siz.set_figwidth(100)#[9]
siz.set_figheight(100)#[9]
#plott.figure(figsize=(100,100)) # [10]
plott.plot()# [10]
```

Figure 4: Source code for the graph generated in figure 3.

III. FINDING THE BEST PATH

If weight can be assigned to the edges on the basis of distance or the time, prim's minimum spanning tree can be generated or the shortest path algorithm given by Dijkstra [6]is very simple to find the best suitable path, for instance rewards for each step is5, it is multiplied by the gamma factor for certain steps if taken. The key attributes of Dijkstra are that the edges are bidirectional and graph is undirected though [9] the directed graph may also be there [7], graph can be traversed on both the directions having weights [7] associated with them. The coding part to implement Dijkstrais not in the scope of the paper, the paper just contributing to the theoretical idea for reinforcement learning that how the oldest algorithm can be beneficial for the new methodologies to reach at best possible decision. The edges as well as nodes have information associated with like weights and ids [7] and generated in python.

Finding path, for example, considering the 5 different paths from graph 2 and their score approximately:

1. 1,2,7,8,19,24,25,29,30=50, ratio=5.5 approx
2. 1,2,7,8,13,18,17,22,27,28,29,30=70, ratio=5.8 approx
3. 1,2,7,8,13,18,17,22,23,28,29,30=125, ratio=10.4 approx
4. 1,2,7,8,19,24,29,30=60, ratio=7.5 approx
5. 1,2,7,8,13,18,23,28,29,30=70, ratio=7 approx

In the above if the score is compared and the ratio with number of steps taken, path 3 is elected as best. Therefore, the graph theory can be beneficial to select the shortest path to gain the maximum score. This way to predict the unknown path optimally, reinforcement learning is best and performed in a better way, Q learning can add the accuracy of the reinforcement learning model. This method is probabilistic method to enhance rewards by considering various possible states and compute their rewards and selecting an optimal path. [12]

IV. DIFFERENCE BETWEEN MACHINE LEARNING, DEEP LEARNING AND REINFORCEMENT LEARNING

Machine learning, deep learning, and reinforcement learning are the fields of artificial intelligence that try to mimic the human brain through continuous learning by experiencing the unknown and un-traversed conditions and deciding the actions to be taken in the future. Machine learning is a broader term of AI i.e. artificial intelligence, where the mechanical instrument is created and programmed to behave intelligently in a real-world environment with minimum or no hazards. Where deep learning or reinforcement learning are the sub-themes or the sub-branches and can be described as the specialized field of machine learning. Deep learning is part of unsupervised learning and the designer doesn't know the outcome; reinforcement learning is aligned between supervised and unsupervised learning. [11] Deep learning is a layered architecture of neural networks where each layer learns from the previous layer. The whole network is trained with some historical data called an example dataset. This example dataset is input to the neural network. Deep learning gives more accurate outcomes, works smoothly with a gradual transition from layer to layer and therefore deep learning is not a replica of the human brain but focused on the accuracy of results and reasoning. In basic terms, reinforcement learning is the part of deep learning that applies to enhance the accuracy of decision-making cumulatively. [1] Reinforcement learning is not a layered-based architecture rather it works on rewards and penalties and the focus of reinforcement learning is to perform in an unseen environment more realistically with no inputs of the historical data or guidelines and to solve a particular problem in its way a like a human brain does. [11]

The major difference between deep learning and reinforcement learning is that deep learning works on an example dataset provided by the designer or the user whereas the reinforcement learning model works from learning via interacting with the environment directly to maximize its rewards. Both models i.e. deep learning and reinforcement

learning model can work collectively in which deep learning incorporates learning experiences and reinforcement learning collects data from the environment to work more intelligently and to face unknown situations.[11] In this way, getting the outcome collectively by two approaches is far better, accurate, and realistic than individual decisions of these two subfields as experiences & results are quite important to learn and to make an improved decision with enhanced creativity. [11]

V. CONCLUSIONS AND FUTURE SCOPE OF WORK

The present paper graphically represents the network containing values and discount factor from rewardstable and explained how the certain path can be selected optimally. Further, the paper added the benefits of discount factor to control the agent behaviour and utility of Q learning in finding a probabilistic path. The chapter also describes the use of Dijkstra algorithm in selecting shortest route, and differences from other learning models. For the future scope of work, back propagation method can aid in selecting the best options at minimum cost.

REFERENCES

[1] guru99. “Reinforcement Learning: What is, Algorithms, Applications, Example”. 2021.https://www.guru99.com/reinforcement-learning-tutorial.html#4

[2] Geeksforgeeks. “Reinforcement learning”. 2020. https://www.geeksforgeeks.org/what-is-reinforcement-learning/

[3] A.I. Wiki. “A Beginner's Guide to Deep Reinforcement Learning”. https://wiki.pathmind.com/deep-reinforcement-learning

[4] Derrick Mwiti. “10 Real-Life Applications of Reinforcement Learning”. 2021.https://neptune.ai/blog/reinforcement-learning-applications

[5] https://www.javatpoint.com/reinforcement-learning

[6] BogoToBogo. “PRIM'S SPANNING TREE ALGORITHM”. Accessed on 2021. https://www.bogotobogo.com/python/python_Prims_Spanning_Tree_Data_Structure.php

[7] Micah Shute. “Dijkstra’s Shortest Path Algorithm in Python”. 2019. https://www.cantorsparadise.com/dijkstras-shortest-path-algorithm-in-python-d955744c7064

[8] Satwik Kansa, Brendan Martin. “Reinforcement Q-Learning from Scratch in Python with OpenAI Gym”. https://www.learndatasci.com/tutorials/reinforcement-q-learning-scratch-python-openai-gym/

[9] Geeksforgeeks. “ML | Reinforcement Learning Algorithm: Python Implementation using Q-learning”. 2019.https://www.geeksforgeeks.org/what-is-reinforcement-learning/

[10] David Landup . “Change Figure Size in Matplotlib”. https://stackabuse.com/change-figure-size-in-matplotlib/

[11] BłażejOsiński and KonradBudek. What is reinforcement learning? The complete guide”. 2018 Jul. Retrieved from: https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/ Date accessed: 01/01/2021

[12] M. Tim Jones. “Models for machine learning”. 2017https://developer.ibm.com/technologies/artificial-intelligence/articles/cc-models-machine-learning/