International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-11 Issue-2, December 2021

Exploration-Exploitation Problem in Policy-Based Deep Reinforcement Learning for Episodic and Continuous Environments

Vedang Naik, Rohit Sahoo, Sameer Mahajan, Saurabh Singh, Shaveta Malik

Abstract: ***Reinforcement learning is an artificial intelligence paradigm that enables intelligent agents to accrue environmental incentives to get superior results. It is concerned with sequential decision-making problems which offer limited feedback. Reinforcement learning has roots in cybernetics and research in statistics, psychology, neurology, and computer science. It has piqued the interest of the machine learning and artificial intelligence groups in the last five to ten years. It promises that it allows you to train agents using rewards and penalties without explaining how the task will be completed. The RL issue may be described as an agent that must make decisions in a given environment to maximize a specified concept of cumulative rewards. The learner is not taught which actions to perform but must experiment to determine which acts provide the greatest reward. Thus, the learner has to actively choose between exploring its environment or exploiting it based on its knowledge. The exploration-exploitation paradox is one of the most common issues encountered while dealing with Reinforcement Learning algorithms. Deep reinforcement learning is the combination of reinforcement learning (RL) and deep learning. We describe how to utilize several deep reinforcement learning (RL) algorithms for managing a Cartpole system used to represent episodic environments and Stock Market Trading, which is used to describe continuous environments in this study. We explain and demonstrate the effects of different RL ideas such as Deep Q Networks (DQN), Double DQN, and Dueling DQN on learning performance. We also look at the fundamental distinctions between episodic and continuous activities and how the exploration-exploitation issue is addressed in their context.***

Keywords: Reinforcement Learning, Episodic Task, Continuous Task, Exploration-Exploitation Problem

I. INTRODUCTION

Reinforcement learning (RL) is the process of learning through the interaction of the agents with their environment.

Manuscript received on November 16, 2021.
Revised Manuscript received on November 19, 2021.
Manuscript published on December 30, 2021.
* Correspondence Author
Vedang Naik*, Department of Computer Engineering, Terna Engineering College, Navi-Mumbai, India. Email: vedangnaik0812@gmail.com
Rohit Sahoo, Department of Computer Engineering, Terna Engineering College, Navi-Mumbai, India. Email: rohitsahoo741@gmail.com
Sameer Mahajan, College of Engineering, Penn State University, Paoli, PA, USA. Email: sam7736@psu.edu
Saurabh Singh, Department of Computer Engineering, Terna Engineering College, Navi-Mumbai, India. Email: srbhsingh39@gmail.com
Dr. Shaveta Malik, Associate Professor, Department of Computer Engineering, Terna Engineering College, Navi-Mumbai, India. Email: shavetamalik687@gmail.com

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)

An RL agent learns from the implications of its actions rather than being directed. It chooses its actions based on previous experiences (Exploration) and potential future paths (Exploitation), effectively hit and miss learning. The RL agent is given a numerical reward based on the success of an action's result, and it attempts to learn to choose actions that maximize the accrued reward over time.

Deep RL is a technique that blends artificial neural networks with a reinforcement learning base so that software agents can learn how to achieve their objectives.

Deep reinforcement learning (deep RL) techniques are obtained when deep neural networks are used to approximate one or more of the following reinforcement learning components: value function, $\hat{v}(s,\theta)$, which is a measurement of how good each state, or state-action combination, is at predicting the likely, cumulative, discounted, future reward. policy $\pi(a|s;\theta)$, which is the agent's behavior, i.e., a mapping from states to actions a and model (state transition function and reward function).

The weights in deep neural networks are the parameters θ in this case; deep learning's capacity enables DRL agents to grow to settings with high-dimensional state and action spaces, overcoming the limitations of conventional RL, which is restricted to relatively low-dimensional issues [1]. In increasingly more complicated, non-stationary settings, deep reinforcement learning agents must communicate, understand, collaborate, synchronize, and contend with other learning agents [2].

The ideal policy selects a "greedy" action that maximizes the value function at each state if the agent knows the proper optimal value function, including the correct estimation of environmental dynamics. If the assessment and prediction are reasonably accurate, a good policy chooses a greedy action known as exploitation.

However, the agent is not aware of the precise optimal value function during the trial-and-error phase. Furthermore, when the environment fluctuates, the value function derived from previous experiences will be suboptimal.

To determine the ideal value function, the agent must perform trial actions, which are not preferable in terms of the present value function; this is referred to as exploration. The below Fig. 1 shows the schema of Deep Reinforcement Learning.

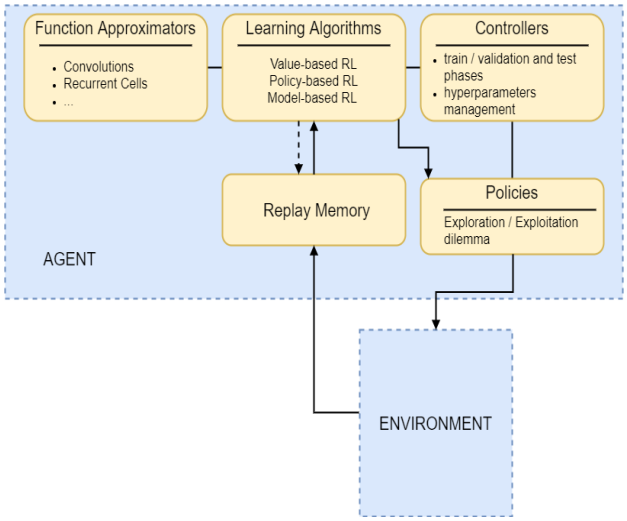

Fig. 1. Deep RL methods schema

The above Fig. 1 shows the schema of Deep Reinforcement Learning methods. The issue is that neither exploration nor exploitation can be done alone without the other faltering. The agent must experiment with various acts and exceedingly favor those that seem to be the optimum. Greedy action has the highest estimated value, and when it is chosen, we say that you are leveraging your current knowledge of the values of the actions. If, on the other hand, you choose one of the non-greedy actions, we call this exploration since it allows you to enhance your estimate of the non-greedy action's value. Exploitation is the best way to maximize the predicted reward on the initial step, while exploration may result in a higher overall reward in the long run. Value-based, policy-gradient, and actor-critic methods are all prevalent ways to train a policy with RL. A value-based method tries to predict potential benefit from states (state value) or action-state (action value or q-value) [3].

The agent–environment interaction naturally breaks down into sub-sequences known as episodes in applications where there is a natural sense of the final time step. The sole reward available in this scenario is the immediate reward. Regardless of how the last episode ended, the following one begins. The next episode starts regardless of how the previous one concluded. As a consequence, the episodes can all be understood to culminate in the same end state, with varied rewards for various outcomes. This type of task is referred to as an episodic task. The termination time is a random variable that fluctuates from episode to episode. On the other hand, ongoing tasks refer to situations in which the agent–environment interaction does not naturally break down into distinct episodes but instead continues indefinitely. In this case, the agent is equally concerned with delayed rewards as it is with immediate rewards.

The paper is organized as follows: Section 2 goes over the environments and the algorithms used. Section 3 goes over the exploration-exploitation dilemma and various strategies employed to tackle it in both episodic and continuous tasks. In Section 4, we see the results of applying the algorithms discussed in Section 2 on episodic and continuous tasks, and Section 5 concludes the paper.

II. ENVIRONMENTS AND ALGORITHMS

A. Episodic Environment

Episodic jobs are those that have a defined endpoint (end). In RL, the interactions between agents and their environments are referred to as episodes. An inverted pendulum, also known as a cartpole, is a pendulum with its center of gravity above its pivot point. It's unstable, but the pivot point under the center of mass may be adjusted to manage it.

The goal is to keep the cartpole in balance by applying appropriate pressures to a pivot point. Repeated attempts to balance the pole are the natural occurrences in this task. A cart travels over a frictionless track and is attached to a pole by an un-actuated joint.

To regulate the mechanism, a force of +1 or -1 is supplied to the cart. When the pendulum begins to stand, the goal is to keep it from falling over. You get a +1 bonus for every timestep the pole stays upright. The episode ends when the pole is tilted more than 15 degrees from vertical or the cart moves more than 2.4 units away from the center.

OpenAI Gym's CartPole-v0 is a basic playground for training and testing Reinforcement Learning algorithms. The gym is a set of tools for creating and testing reinforcement learning systems. The gym library is a set of test problems (environments) that you may use to evaluate your reinforcement learning systems.

B. Continuous Environment

There is no such thing as a terminal state in a continuous task. Reinforcement learning may be applied to stock price prediction since it follows the same principles of using less previous data and operating in an agent-based system to forecast greater returns depending on the present environment. Because of its dynamic nature, learning just from past data is insufficient; the model must learn continually.

Stock price prediction is a challenging task since it moves quickly and data is frequently limited. DRL would help determine the best solutions to such complex decision problems, and it may be a better option for stock price prediction and enhance expected return. In addition, deep learning algorithms are capable of extracting features from raw data with many parameters. The data comes from Yahoo Finance and includes the price history and trading volumes of Google stocks from 2011 through 2021.

C. Algorithms

In our experiment, we used Deep Q-Network (DQN), Double Deep Q-Network (DDQN), and Dueling Deep Q-Network (Dueling DQN) for continuous tasks to automate stock price prediction as well as for episodic tasks to keep the pendulum start vertical and prevent it from collapsing. To learn the parameters θ of the policy function $\pi\theta(at|st)$, a plethora of reinforcement learning algorithms are applicable [4].

Retrieval Number: 100.1/ijeat.B32671211221
DOI: 10.35940/ijeat.B3267.1211221
Journal Website: www.ijeat.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

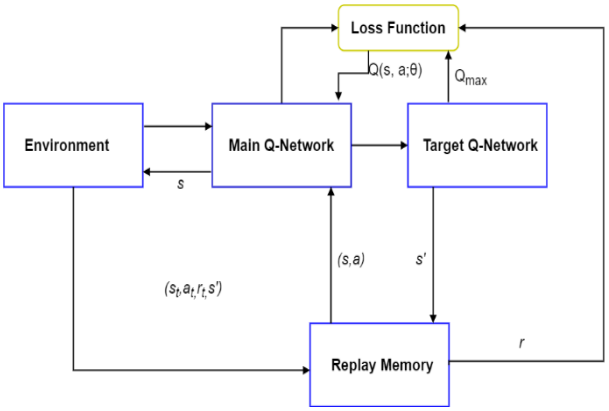

Fig. 2. Deep Q Network Model Structure

A. Deep Q-Network

It's a system for sequential decision-making that uses reinforcement learning. Learning has as its goal the discovery of the optimum policy that maximizes long-term advantages. The agent is rewarded rt for taking action depending on the present condition st of the environment. The experience replay is utilized to remember former states, actions, rewards, and forthcoming states to understand previous events. Data from the replay memory is randomly selected and sent to the train network in small batches to minimize the error. DQN adds up all activities that contribute to an overstatement of the Q-value as the number of iterations increases and the mistakes accumulate [5]. In DQN, the Q network is leveraged to learn the projected future reward Q-value function (Q(st, at)). The Deep Q-Network differs from the standard Q-learning algorithm because it uses a new Target-Q-Network, which is provided by:

$$Q_{target} = r_{t+1} + \gamma \max a' [Q(s_t, a_t ; \theta)]$$

Where parameters of Q network are defined by θ. The target Q-Network is distinct from the general Q-Network, which is continually modified. The network values of the target Q-Network are continuously updated and duplicate the values of the main network. When the incoming data rate is high, and the training data is strongly correlated, using only one Q-Network in the model results in delayed or sub-optimal convergence, as well as an imbalanced target function. Using two distinct Q-Networks improves the Q-stability to choose and evaluate the actions; the ideal Q-value or action-value pair is determined. DQN adds up all activities that contribute to an overstatement of the Q-value since mistakes add up as the number of epochs increases [6]. The Double DQN neural network solves the overstatement of Q-value by using another neural network that optimizes the impact of inaccuracy.

B. Double Deep Q-Network

When actions are made based on a Target Q-Network, the problem of overstatement becomes more significant since the Target Q-values of the network are not continually updated. Because it gives additional stability to the target values for updates, Double DQN utilizes two neural networks with the same structure as DQN, the main network, and the target network. The action in DDQN is picked according to the main Q network, yet it employs the target state-action value. This value comes from the Target Q network that matches to that specific state-action. Thus, all action-value pairs for all possible actions in the current state are upgraded at each time interval and step.

$$Q_{target} = r_{t+1} + \gamma Q(s_t, \text{argmax } a' Q(s_t, a_t ; \theta); \theta')$$

C. Dueling Double Deep Q-Network

When compared to DQN, Dueling DQN has a different network topology. Dueling DQN employs a customized Dueling Q Head to split Q into an A (advantage) and a V stream. Incorporating this sort of structure to the network head helps it distinguish actions from each other, which increases learning substantially.

In many situations, the rewards of many actions are relatively comparable, making it less essential which action to perform. This is especially crucial in cases where there are several options. In DQN, we update the Q values solely for the particular actions done in those states throughout each training step for every state in the batch. This causes delayed learning since we do not learn the Q values for activities that have not yet been done [7]. Learning on dueling architecture is quicker since we begin learning the state-value; a single action has been done at this stage.

III. EXPLORATION-EXPLOITATION DILEMMA

Exploration and exploitation are two critical ideas in reinforcement learning. However, exploration and exploitation are two mutually exclusive components. Exploration is the process of selecting activities that have never been done before to discover new possibilities [8]. Exploitation refers to the selection of activities taken to improve the models of known actions and to make use of previously acquired information [9]. The difficulty is that determining the best long-term plan may necessitate making short-term compromises, and making the best overall judgments typically necessitates having sufficient knowledge. In RL, finding a balance between exploration and exploitation is a major topic and difficulty. Many techniques have been explored in previous studies to make this trade-off:

Greedy Policy: The agent will do the action with the highest Q value at any point in the environment. This is a naïve way of guaranteeing that the agent performs the best possible action at each stage. The method's apparent flaw is that the agent will never witness the consequence of any action other than the ideal one. This invariably results in a less-than-ideal answer.

Epsilon-Greedy Policy: This is the most frequently used exploration approach, and it consists of a mix of random operations with probability and greedy operations when not. By traveling to an unknown area of the state space, the agent can conduct actions that give new knowledge. The variable is changeable, and it is usually set to a high value at first, then gradually reduced to a low one. Despite its widespread use, this technique is far from ideal, as it simply considers whether activities are most beneficial or not.

Boltzmann Policy: Rather than constantly picking the best approach or acting randomly, this method includes selecting a course of action based on weighted probability.

© Copyright: All rights reserved.

This is achieved by applying a SoftMax to all of the Q values at any given state. As a result, the agent is more likely to choose the action that is most likely to be optimum. The plurality of RL algorithms employs a point estimate of the Q-function to determine such an optimum policy Q(s, a) [10].

$$\pi(a|s) = \frac{exp(Q(s,a)/\boldsymbol{\tau})}{\sum_{i=1}^{n} exp(Q(s,i)/\boldsymbol{\tau})}$$

A. Cartpole Problem

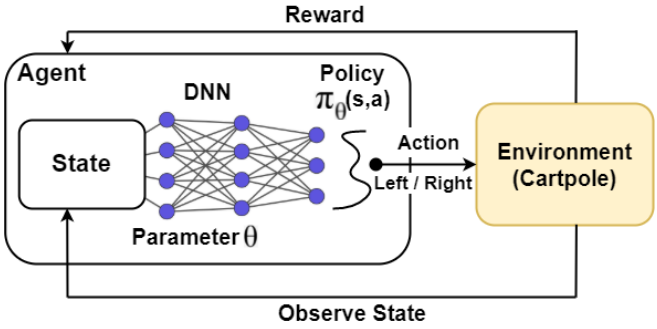

Fig. 3. Cartpole problem model structure

The above Fig. 3 shows the model structure of Cartpole problem. The state vector for this system x is a four-dimensional vector with components {x, x', θ, θ'}. There are two states to the action: left (0) and right (1). The episode ends when

- the pole angle is higher or lesser than 12 degrees from the vertical axis
- the cart location is greater than 2.4 cm from the center
- the episode duration is greater than 200

The agent is rewarded one point for each step completed, including the termination step. The issue is declared solved if the average reward across 100 episodes is more than or equal to 195. Exploration for this problem involves moving the cart in a random direction and noting the results of this action. In contrast, exploitation involves taking into account the current state of the pole and then acting in a way that is appropriate based on the strategy chosen from above.

B. Stock Market Prediction

Stock price prediction is a difficult undertaking since the stock market moves fast and data is sometimes inadequate or insufficient [11]. One approach for resolving such complicated choice issues is reinforcement learning. The state, action, and reward are defined in this section, as well as how we represent the stock trading issue as an MDP.

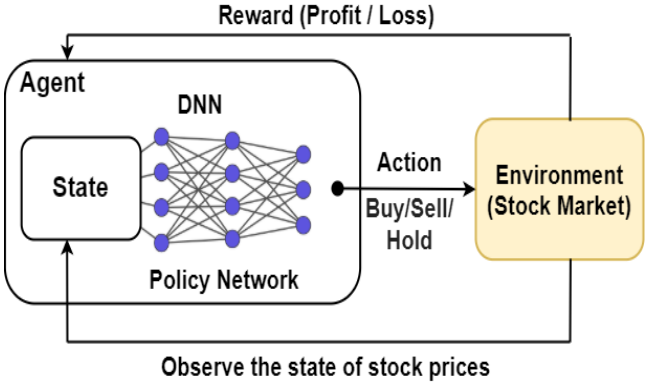

Fig. 4. Stock Market prediction model structure

- **State s:** the state is defined as the daily adjusted closing data.

St = [Pt-20, Pt- l9, ..., Pt-2, Pt-l]

pt represents the adjusted close price of day t.

- **Action a:** Only three values [1,0, -1] are accessible in the action space, indicating purchase, do nothing, and sell for one stock, respectively. We also define taking an action as trading one share of stock with the prices of the current day's adjusted close and automatically closing the position with the price of the following day's adjusted close because we focus on the daily trading process.
- **Reward r:** the reward is calculated by the difference between next day's adjusted close and the current day's adjusted close.

rt = (pt+l - pt) * action – value

We start by initializing the parameters of Q-networks with a modest batch of training data. The state is then utilized as the input to the Q-network on each trading day, with the Q values for three possible actions as the output. The agent will be rewarded if he picks the action with the highest value with a probability of (1 - e).

The trade sample (s, a, r, s') will be saved in the buffer after that. Because the daily training sample is very tiny for a deep neural network, overfitting is a possibility. We keep three samples with three distinct rewards in the buffer to enrich the data used for updating the Q-network parameters and make the updating more reliable.

After that, the total reward is computed, and a random number of batch-size samples from the buffer are taken to update the Q network settings. The entire procedure will be repeated until the conclusion of the test data, at which point a cumulative reward will be acquired.

In this problem, exploration is done by purchasing, selling, or keeping a stock at random, but exploitation is done by using prior experience and behaving in a way that maximizes long-term profit.

IV. RESULTS AND PERFORMANCE

Exploration-Exploitation in Episodic Task

OpenAI Gym's CartPole-v1 is used for training and testing three deep Q-networks.

The dataset is trained for 5000 steps. To approximate Q, one traditional method is creating a lookup table where the values of Q are updated after each agent's actions.

However, this approach is slow and does not scale to significant action and state spaces. Since neural networks are universal function approximators, we will train a network that can approximate Q.

A simple neural network was implemented using TensorFlow Keras, having four dense layers with activation function as ReLU. The network takes the agent's state as an input and returns the Q values for each action. The agent selects the maximum Q value to perform the following action.

Retrieval Number: 100.1/ijeat.B32671211221
DOI: 10.35940/ijeat.B3267.1211221
Journal Website: www.ijeat.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

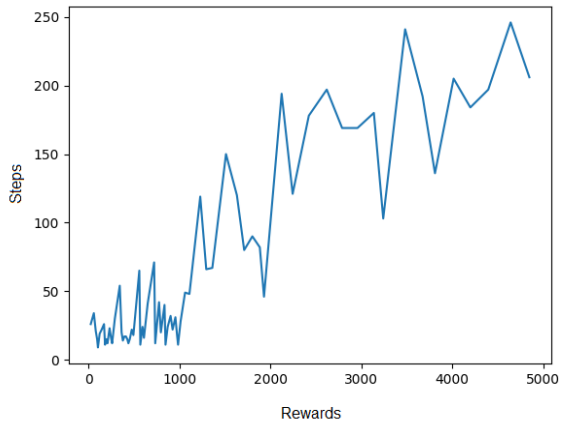

Fig. 5. DQN Cartpole

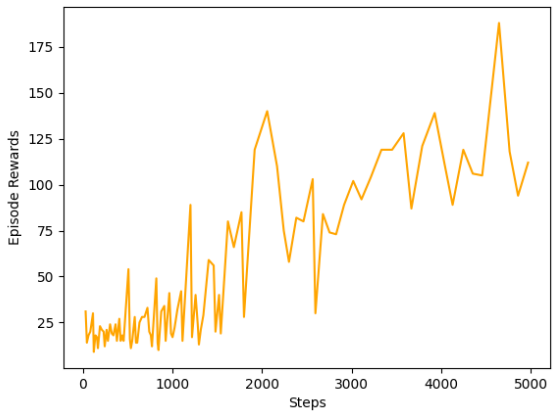

Fig. 6. Double DQN Cartpole

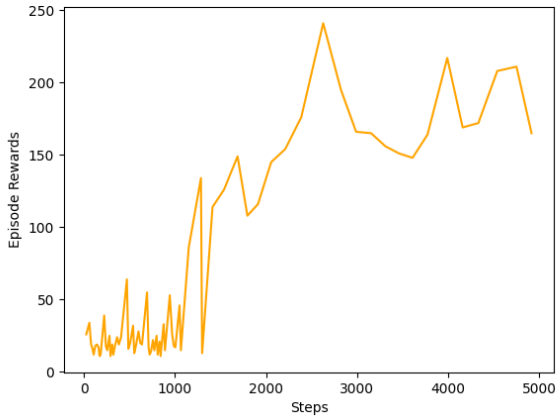

Fig. 7. Dueling DQN Cartpole

The cartpole is tested for various policies such as Greedy Q policy, Epsilon Greedy Q policy, and Boltzmann policy. Fig.5, Fig. 6 and Fig. 7 shows the training rewards for number of steps with the help of DQN, Double DQN, and Dueling DQN. The training is done for 5000 steps and testing is done for five episodes. The testing rewards can be seen in Table 1; the reward shows the best reward among the five testing episodes.

Table 1. Cartpole Results

	Reward	Training Steps	Testing Episodes
DQN	459	5000	5
DDQN	383	5000	5
Dueling DQN	500	5000	5

Exploration-Exploitation in Continuous Task

The studies are carried out using Google stock datasets and three DQN-based RL Algorithms. These algorithms are used to determine the total rewards and profit from training and testing data for Google stocks. Training data was collected for 2400 days to understand how the algorithm performs, and test data was collected for 117 days. Then after using all the data, we can predict for the next day by moving the window, where the window length is of 5 days, i.e., five days of data is used to predict the next day. The Google stock dataset is used to illustrate the training rewards about the steps it was trained for. The testing rewards can be seen in Table 1, the total profit and reward that was made for the next day after the total days in the dataset.

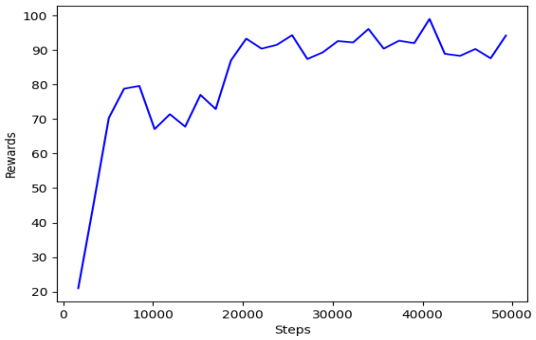

Fig. 8. DQN Stock Market Prediction

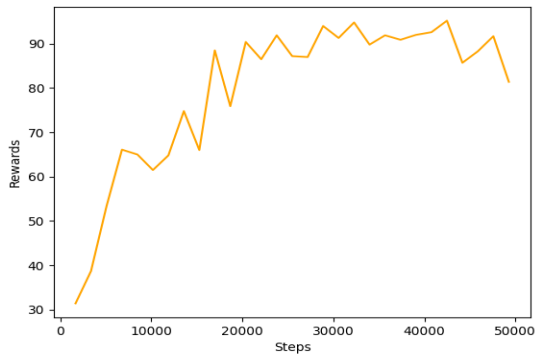

Fig. 9. DDQN Stock Market Prediction

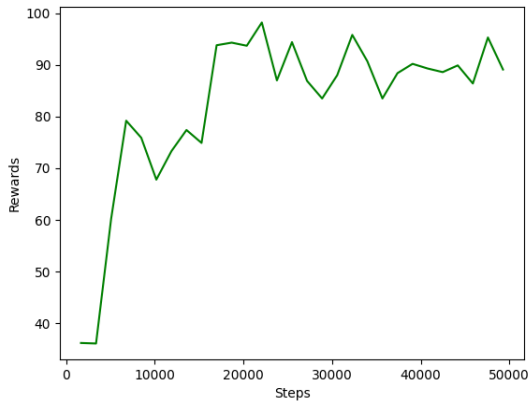

Fig. 10. Dueling DQN Stock Market Prediction

© Copyright: All rights reserved.

Table 2. Stock Market Results

	Reward	Total Profit	Time steps	Profit %
DQN	82.3689	1.06	20000	6.20
DDQN	79.6069	1.018	20000	1.80
Dueling DQN	80.34483	1.038	20000	3.80

V. CONCLUSION AND FUTURE SCOPE

In this paper, we give an introduction to the exploration-exploitation problem and episodic and continuous tasks. We touched on various forms of Deep Q Networks. Also, we have looked at the exploration-exploitation problem in the context of continuous and episodic tasks of stock market trading and the cartpole problem, respectively and performed a comparative analysis on them. In the future, we will implement other reinforcement algorithms such as A2C, DDPG, PPO to understand more about exploration-exploitation problems in different environments.

REFERENCES

1. G. Sokar, E. Mocanu, D. C. Mocanu, M. Pechenizkiy, and P. Stone, Dynamic Sparse Training for Deep Reinforcement Learning. 2021.
2. P. Danassis, A. Filos-Ratsikas, and B. Faltings, Achieving Diverse Objectives with AI-driven Prices in Deep Reinforcement Learning Multi-agent Markets. 2021.
3. J. Ault and G. Sharon, "Reinforcement Learning Benchmarks for Traffic Signal Control," 2021. [Online]. Available: https://openreview.net/forum?id=LqRSh6V0vR
4. S. Messaoud, M. Kumar and A. G. Schwing, "Can We Learn Heuristics for Graphical Model Inference Using Reinforcement Learning?," 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 7586-7596, doi: 10.1109/CVPR42600.2020.00761.
5. H. van Hasselt, A. Guez, and D. Silver, Deep Reinforcement Learning with Double Q-learning. 2015.
6. T. Matiisen, Institute of Computer Science, University of Tartu, https://neuro.cs.ut.ee/demystifying-deep-reinforcement-learning/ (accessed: 2015).
7. S. Ishii, W. Yoshida, and J. Yoshimoto, "Control of exploitation–exploration meta-parameter in reinforcement learning," *Neural Networks*, vol. 15, no. 4, pp. 665–687, 2002.
8. R. Sutton and A. Barto, Reinforcement Learning: An Introduction, Zweite. MIT Press, 2018.
9. N. Nikolov, J. Kirschner, F. Berkenkamp, and A. Krause, Information-Directed Exploration for Deep Reinforcement Learning. 2019.
10. J. W. Lee, "Stock price prediction using reinforcement learning," in ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No.01TH8570), 2001, vol. 1, pp. 690–695 vol.1. doi: 10.1109/ISIE.2001.931880.

AUTHORS PROFILE

Vedang Naik is an engineer with a bachelor's degree in Computer Engineering from the Terna Engineering College, University of Mumbai, Navi-Mumbai, India. His research interests are in Deep Learning, Natural Language Processing, Neural Networks and Machine Learning.

Rohit Sahoo is an engineer with a bachelor's degree in Computer Engineering from the Terna Engineering College, University of Mumbai, Navi-Mumbai, India. His research interests are in Machine Learning, Deep Learning, Data Science and Big Data.

Saurabh Singh is an engineer with a bachelor's degree in Computer Engineering from the Terna Engineering College, University of Mumbai, Navi-Mumbai, India. His research interests are in Data Mining, Machine Learning, Recommender Systems, and Data Science.

Sameer Mahajan is currently pursuing master's degree in Data Analytics from the Penn State University, Pennsylvania, USA. His research interests are in Differential Privacy, Machine Learning, Federated Learning, and Data Science.

Dr. Shaveta Malik is working as Associate Professor at Computer Engineering Department at Terna Engineering College, Navi-Mumbai, India. She has presented several papers in international conferences. Her research interests are in Artificial Intelligence, Machine Learning and Image Processing.

© Copyright: All rights reserved.