CartPole Balancing using ReinforcementLearning

Raghav Khanna
Computer Science & Engineering
Lovely Professional University
Jalandhar, India
raghav.12008119@lpu.in

I. Abstract

This paper presents a comprehensive exploration of training a CartPole balancing bot using Q-learning, within the OpenAI Gym environment. The CartPole problem serves as a quintessential challenge in reinforcement learning, where the agent must learn optimal strategies to balance a pole atop a moving cart.

Theoretical foundations of Q-learning are discussed, providing insight into the algorithm's suitability for dynamic decision-making scenarios. Implementation details of the training process, encompassing the CartPole environment setup, Q-learning algorithm specifics, and training parameters, are thoroughly examined.

Experimental results showcase the agent's learning curve and its ability to achieve equilibrium in the face of varying conditions. Insights gained from this study contribute to a deeper understanding of reinforcement learning's applicability in dynamic and complex scenarios.

II. Introduction

The CartPole balancing problem serves as a fundamental benchmark in the field of reinforcement learning, specifically within the OpenAI Gym environment. This challenge involves training an agent to balance a pole on a moving cart, demonstrating the agent's ability to learn optimal strategies for maintaining equilibrium.

Reinforcement learning, a subset of machine learning, is particularly well-suited for solving dynamic decision-making problems, making it an apt approach for training agents in scenarios like the CartPole environment. In this context, Q-learning stands out as a powerful algorithm for reinforcement learning tasks, providing a foundation for our exploration.

This paper aims to delve into the intricate process of training a CartPole balancing bot using Q-learning. By understanding the underlying principles, implementation details, and experimental outcomes, we can gain insights into how reinforcement learning algorithms contribute to the development of intelligent systems capable of dynamic decision-making.

In the subsequent sections, we will discuss the theoretical underpinnings of Q-learning, the specifics of the CartPole environment, the implementation details of our training approach, and finally, the results and insights gained from our experiments.

Mastering the CartPole balancing problem is not only a captivating endeavor in the realm of reinforcement learning but also a

stepping stone toward comprehending the broader challenges of training agents for real-world applications. The fusion of theory and practical implementation in this exploration sets the stage for unraveling the potential of Q-learning and reinforcement learning as a whole.

III. Methodology:

1. Problem Definition:
   - The study focuses on training a CartPole balancing bot using Q-learning in the OpenAI Gym environment. The primary goal is to develop an agent capable of effectively balancing a pole on a moving cart.

2. Environment Setup:
- The CartPole environment in OpenAI Gym is configured with specific parameters, including cart mass, pole length, and gravitational acceleration, ensuring a standardized setup for experimentation.

3. Q-learning Algorithm:
- The Q-learning algorithm is detailed, covering key components such as state representation, available actions, immediate rewards, and the Q-value update process. The initialization and updating of the Q-table or Q-network are explained.

4. Training Procedure:
- The training process is outlined, including the exploration-exploitation strategy employed by the agent. Convergence criteria are established, and the number of training episodes required for the agent to learn optimal policies is determined.

5. Hyperparameter Tuning:
- Hyperparameters such as learning rate, discount factor, and exploration rate are selected and tuned. The rationale behind the chosen values is discussed, and any sensitivity analysis performed is highlighted.

6. Data Collection:
- The process of collecting training data during each episode is explained, addressing any challenges or considerations in data collection to ensure the quality of the dataset.

7. Data Analysis:
- The collected data is analyzed using appropriate statistical or qualitative methods. This step involves interpreting the results and drawing conclusions based on the performance metrics.

Q-learning is a type of reinforcement learning algorithm that learns the value of taking an action in a given state. It does not need a model of the environment, and it can deal with uncertain outcomes and rewards. Q-learning aims to find the optimal policy that maximizes the expected value of the total reward over any and all future steps, starting from the current state. [1]

Q-learning works by using a table, called the Q-table, that stores the estimated value of each state-action pair. The value of a state-action pair, also known as the Q-value, represents how good it is to take that action in that state. The Q-value is updated iteratively using the Bellman equation, which incorporates the immediate reward and the discounted future reward. The Q-learning algorithm explores the environment by taking actions and observing the rewards and the next states. It then updates the Q-table based on the difference between the old and the new Q-value, also known as the temporal difference error. The exploration strategy can be random or guided by some heuristics, such as the epsilon-greedy method, which chooses a random action with a small probability and the best action with a high probability. The Q-learning algorithm converges to the optimal Q-table as long as each state-action pair is visited infinitely often and the learning rate is appropriately chosen. [2]

Q-learning can be applied to a variety of problems, such as controlling a robot, playing a game, or navigating a maze. For example, in the Cart-Pole problem, the agent has to balance a pole on a cart by moving left or right. The state can be defined by the position and velocity of the cart and the angle and angular velocity of the pole. The action can be either applying a positive or a negative force to the cart. The reward can be either a constant positive value for each time step that the pole remains upright, or a negative value for each time step that the pole falls over. The Q-learning algorithm can learn to balance the pole by trying different actions and updating the Q-table based on the rewards and the next states. [3]

IV. **Software and Hardware Setup**

The complete implementation code for this paper is avail- able on GitHub []. The program is written using Python. It takes about a couple of minutes for running about 90000 episodes on a HP Omen laptop with a Nvidia GeForce RTX 1650ti GPU card with 4 GB of video ram. It is also possible to make use of freely available GPU cloud such as Google Colab or Kaggle

V. **Metrics used:**

$$Q(st,at)=Q(st,at)+\alpha[rt+1+\gamma amaxQ(st+1,a)-Q(st,at)]$$

where:

- st is the current state
- at is the action taken
- rt+1 is the reward observed after taking the action
- α is the learning rate
- γ is the discount factor
- maxaQ(st+1,a) is the maximum Q value over all possible actions in the new state[23].

Learning Rate (α): The learning rate, often referred to as alpha or α, determines how much newly acquired information overrides old information12. A factor of 0 makes the agent learn nothing (exclusively exploiting prior knowledge), while a factor of 1 makes the agent consider only the most recent information (ignoring prior knowledge to explore possibilities)1. It’s how big you take a leap in finding an optimal policy3.

Discount Factor (γ): The discount factor, often denoted as gamma or γ, determines the importance of future rewards. A discount factor of 0 will make the agent “short-sighted” by only considering current rewards, while a discount factor of 1 will make it strive for a long-term high reward4567. It’s used to balance the trade-off between immediate and future reward.

Epsilon (ε): In the context of epsilon-greedy policy in reinforcement learning, epsilon refers to the probability of choosing to explore891011. The agent will choose exploration with probability ε and will exploit current knowledge with probability 1-ε891011.

Epsilon Decay: Epsilon decay is a method used to gradually reduce the value of epsilon over time3121314. This means that the agent will explore a lot in the beginning and slowly shift to exploiting its learned policy as it becomes more confident3121314.

```
# Divide position, velocity, pole angle, and pole angular velocity into segments

    pos_space = np.linspace(-2.4, 2.4, 5)

    vel_space = np.linspace(-4, 4, 5)

    ang_space = np.linspace(-.2095, .2095, 5)

    ang_vel_space = np.linspace(-4, 4, 5)



learning_rate_a = 0.3 # alpha or learning rate

    discount_factor_g = 0.95 # gamma or discount factor.

  epsilon = 1         # 1 = 100% random actions

    epsilon_decay_rate = 0.00001 # epsilon decay rate
```

These parameters are crucial in balancing exploration (trying out new actions to improve future decisions) and exploitation (using current knowledge to make the best decision now) in reinforcement learning891011.

```
# Libraries used

import gymnasium as gym

import numpy as np

import matplotlib.pyplot as plt

import pickle

import os
```

VI. **Results:**

The training of the CartPole balancing bot using Q-learning has yielded noteworthy outcomes, as evidenced by the progression of episodes and corresponding rewards. A comprehensive analysis of key milestones and trends in the results is presented below:

1. Training Progression:

   - The training spanned episodes from 87500 to 99600, showcasing a dedicated effort to refine the agent's performance.

2. Reward Fluctuations:

   - The rewards achieved per episode varied, indicating the adaptability of the Q-learning algorithm in responding to dynamic challenges.

   - Episodes with lower rewards (e.g., Episode 87600 with 14.0) demonstrate instances where the agent faced difficulties in maintaining balance.

   - Conversely, episodes with higher rewards (e.g., Episode 99600 with 169.0) reflect significant improvement, showcasing the learned strategies of the agent.

3. Epsilon Decay:

   - The epsilon values accompanying each episode illustrate the exploration-exploitation trade-off. As training progresses, epsilon gradually decreases, emphasizing a shift from exploration to exploiting the learned policies.

4. Mean Rewards:

- The mean rewards across episodes provide a consolidated view of the agent's overall performance.

- Variations in mean rewards suggest fluctuations in the agent's proficiency, with certain episodes demonstrating improved balancing capabilities (e.g., Episode 99600 with Mean Rewards 266.2).

5. Convergence Indicators:

- The sustained exploration in earlier episodes is reflected in instances where the agent's mean rewards rise and fall (e.g., Episode 87700 to Episode 88000).

- Convergence is achieved as seen in the later episodes, where the mean rewards stabilize at higher values, indicating the agent's ability to consistently balance the pole (e.g., Episode 99600).

6. Q-Table Persistence:

- The persistence of the Q-table, saved as 'Qtable.pkl,' provides a valuable resource for understanding the learned policies and potential applications beyond the training phase.

7. Post-Training Evaluation:

- The subsequent execution of the trained agent in a new environment (as indicated in the script output) showcases the transferability of learned policies to novel scenarios.

The results depict a commendable progression in the agent's ability to balance the CartPole, reaffirming the effectiveness of Q-learning in mastering dynamic decision-making tasks. The observed fluctuations underscore the iterative nature of reinforcement learning, emphasizing the significance of sustained training for achieving optimal performance. The saved Q-table opens avenues for further exploration and potential real-world applications of the learned policies.

VII. **Conclusion:**

In conclusion, the endeavor to train a CartPole balancing bot using Q-learning has proven to be a dynamic exploration marked by significant achievements. The progression of the agent through episodes, the nuanced fluctuations in rewards, and the epsilon decay collectively showcase the adaptability and learning capabilities of the Q-learning algorithm in the face of the challenging CartPole environment.

The observed convergence in mean rewards, particularly in the later episodes, signals the successful acquisition of policies that enable the agent to consistently balance the pole. This not only emphasizes the potency of reinforcement learning but also underscores the iterative nature of training, where the agent refines its strategies over time.

The persistence of the Q-table, symbolized by the saved 'Qtable.pkl' file, holds promise for future applications. The learned policies encapsulated in the Q-table could be leveraged in diverse scenarios beyond the training environment, showcasing the transferability and practical utility of the trained agent.

As the agent achieves stability and high mean rewards, it reflects a mastery of the CartPole balancing task, affirming the effectiveness of Q-learning in addressing real-world dynamic decision-making challenges. The observed fluctuations, both in rewards and epsilon values, highlight the delicate balance between exploration and exploitation, a crucial aspect of reinforcement learning.

In essence, this study contributes not only to the understanding of Q-learning in the context of CartPole balancing but also lays the groundwork for further exploration and application of reinforcement learning techniques in diverse and complex scenarios.

VIII. **References**

[1] OpenAI Gym, Toolkit for developing and comparing reinforcement learning algorithms. https://gym.openai.com/.

[2] A. Gulli and S. Pal. *Deep learning with Keras*. Packt Publishing Ltd, 2017.

[3] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[4] G. E. Hinton, "Connectionist learning procedures," Artificial Intelligence, vol. 40, no. 1–3, pp. 185–234, September 1989.

[5] S. Kumar, (2020) Balancing a CartPole System with Reinforcement Learning – A Tutorial arXiv:2006.04938v2 [cs.RO].