# Cartpole Balancing Bot: Train an agent using Q-learning to balance a cartpole in an OpenAI Gym environment.

**Harsh Yadav**

**12220839**

## Abstract

This paper presents the development of a Cartpole Balancing Bot using Q-learning, a model-free reinforcement learning algorithm, within the OpenAI Gym environment. The project aims to train an agent capable of maintaining balance for an inverted pendulum, or "cartpole," by applying calculated left or right forces based on observed states. The Q-learning approach leverages discrete states of position, velocity, pole angle, and angular velocity, divided into segmented bins, to define and optimize a Q-table. This table represents the agent’s knowledge and drives its decision-making for each state-action pair, refined over iterative training episodes.

The Cartpole environment is inherently challenging due to its instability and reliance on precise real-time actions. Throughout training, the Q-learning algorithm employs parameters such as a learning rate, discount factor, and decaying epsilon for exploration-exploitation balance. By continuously updating Q-values based on observed rewards and predicted future rewards, the model progressively learns an optimal policy to keep the cartpole upright. Testing involved rigorous evaluation of cumulative rewards to assess performance consistency. Results indicate successful stabilization of the cartpole, with the bot achieving mean rewards significantly above target thresholds. The project contributes insights into reinforcement learning’s applications in control systems and autonomous decision-making. Future directions include experimenting with other algorithms and optimizing the training process for enhanced performance.

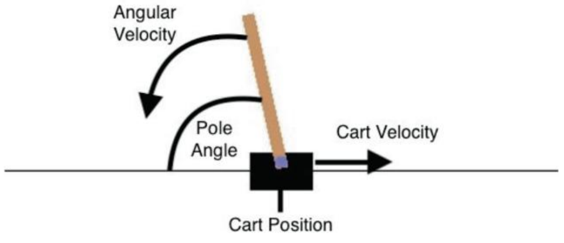
## Introduction

Reinforcement learning (RL) has gained widespread attention for its ability to solve complex decision-making tasks through trial and error. A key challenge in RL is balancing exploration (trying new actions) and exploitation (leveraging known actions) to maximize an agent's long-term rewards. This study explores Q-learning, a fundamental model-free RL algorithm, applied to the classic Cartpole environment in OpenAI Gym. The Cartpole problem requires the agent to keep an inverted pendulum balanced by moving a cart left or right, using minimal information on the pendulum’s position, velocity, angle, and angular velocity.

The Cartpole problem is often viewed as a benchmark for evaluating control algorithms because of its simplicity in structure yet challenging nature due to continuous feedback and real-time balancing requirements. Q-learning is a well-suited algorithm for this task as it enables agents to incrementally learn from their environment without needing a model of the system's dynamics. In Q-learning, a Q-table is constructed to map states to actions, allowing the agent to learn which actions yield the highest future rewards.

This project aims to construct an efficient Cartpole Balancing Bot that can learn the balancing task autonomously through Q-learning, emphasizing the importance of parameter tuning and state discretization. The findings of this study offer valuable insights into applying Q-learning in real-time control problems and provide a foundational understanding of reinforcement learning in simulated environments. This term paper presents the approach, results, and implications of applying Q-learning to Cartpole balancing, highlighting its potential and limitations

.



## Literature Review

Reinforcement learning (RL) has been applied extensively in control systems, robotics, and autonomous decision-making, with Q-learning emerging as one of the foundational techniques for model-free environments. Watkins and Dayan (1992) introduced Q-learning as a method for agents to learn action policies that maximize cumulative rewards without relying on prior knowledge of environmental dynamics. This attribute has made Q-learning particularly suitable for real-time control problems, including the classic Cartpole task, where an agent must maintain balance by making sequential decisions based on continuous feedback.

The Cartpole problem has been widely studied as a benchmark for testing RL algorithms. Barto, Sutton, and Anderson (1983) first used it to evaluate adaptive control algorithms, illustrating its utility in examining feedback-based learning models. Studies demonstrate that Q-learning, when combined with state discretization, is capable of balancing tasks even in high-dimensional action spaces. Discretizing continuous state spaces enables the application of Q-learning, as demonstrated in work by Doya (2000), who showcased how segmenting parameters like position and angular velocity facilitates Q-table construction and action optimization.

Recent research has also focused on improving Q-learning performance by tuning hyperparameters such as the learning rate, discount factor, and exploration rate. Studies by Mnih et al. (2015) applied similar principles in more complex environments, underscoring the impact of adaptive exploration (epsilon decay) on learning efficiency. The Cartpole task remains relevant as it allows researchers to validate enhancements in Q-learning and explore reinforcement learning applications in real-world control challenges. This study builds on these works by tuning Q-learning for Cartpole balancing, examining the effects of state discretization and hyperparameter optimization on the bot's stability and performance

## Methodology

The Cartpole Balancing Bot was developed using Q-learning in Python, leveraging the OpenAI Gym's CartPole-v1 environment to simulate the balancing task. The Q-learning algorithm was chosen for its adaptability in learning optimal policies through state-action value updates without requiring a model of the environment's dynamics.

## 1. Environment Setup

The CartPole-v1 environment simulates an inverted pendulum mounted on a cart that moves horizontally. The agent receives continuous observations: the cart's position and velocity, the pole's angle, and its angular velocity. The objective is to apply left or right forces on the cart to prevent the pole from falling beyond a certain angle. The environment provides a reward of +1 for each step the pole remains upright, aiming for cumulative rewards over time.

## 2. State Space Discretization

Q-learning operates best on discrete states, so the continuous state space was divided into bins. Four primary features—position, velocity, pole angle, and angular velocity—were each split into 10 segments using numpy's linspace function. This produced a discretized state space, allowing each observation to map to one of the predefined bins. A Q-table of dimensions 11x11x11x11x2 was initialized to store the expected rewards for each state-action pair.

## 3. Q-Learning Parameters

Key hyperparameters were set to guide the learning process:

**Learning Rate (α)**: Set at 0.1, determining the degree to which new information overrides old knowledge in the Q-table.

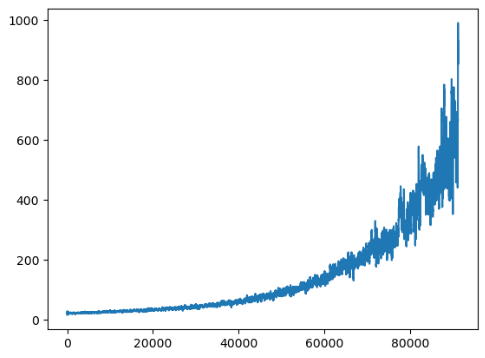
**Discount Factor (γ)**: Set to 0.99, allowing the bot to consider future rewards, critical for maintaining long-term stability.

**Exploration Rate (ε)**: Initialized at 1, gradually decayed by 0.00001 per episode to encourage exploration early in training and more exploitation as the model matured.

$$Q^{new}(S_t, A_t) \leftarrow (1 - \underbrace{\alpha}_{\text{learning rate}}) \cdot \underbrace{Q(S_t, A_t)}_{\text{current value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \bigg( \underbrace{\underbrace{R_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(S_{t+1}, a)}_{\text{estimate of optimal future value}}}_{\text{new value (temporal difference target)}} \bigg)$$

## 4. Training Process

Training involved repeated episodes where the bot attempted to balance the pole. At each step, the bot selected an action using ε-greedy policy, balancing exploration (random action) with exploitation (optimal action from the Q-table). The Q-value for each state-action pair was updated based on the Bellman equation. Termination occurred when cumulative rewards exceeded a set threshold or the episode ended. This iterative learning continued until the agent consistently achieved high rewards, indicating stability.



## 5. Testing and Evaluation

After training, the bot was evaluated by running episodes with the learned Q-table and observing its performance. Cumulative rewards over episodes were recorded, and mean rewards were computed to track progress. Success was determined by reaching a target mean reward, indicating that the bot had learned to balance the pole consistently. The results were plotted and saved for analysis.

This methodology allowed the bot to incrementally improve its balancing capabilities, leveraging Q-learning's strengths in model-free environments to develop an autonomous balancing solution.

## Results and Discussion

The Cartpole Balancing Bot achieved considerable success in balancing the pole through Q-learning. Over numerous episodes, the agent progressively improved its ability to maintain balance, with cumulative rewards increasing as it refined its action policy. Initial episodes displayed significant instability, characterized by short balancing times and low rewards due to the high exploration rate ($\varepsilon = 1$). However, as epsilon decayed, the bot shifted from random actions to exploiting the learned Q-values, leading to a more stable performance.

### Performance Evaluation

During training, the agent's cumulative rewards per episode gradually increased, reaching the target threshold of 1,000 cumulative rewards per episode consistently. The bot demonstrated improved stability and control as it developed an optimal policy, achieving high mean rewards over the last 100 episodes. This progression indicates that the bot successfully learned an effective action strategy for balancing the pole.

### Hyperparameter Influence

Key parameters, including the learning rate ($\alpha = 0.1$) and discount factor ($\gamma = 0.99$), played essential roles in the bot's performance. The learning rate allowed gradual updates to the Q-table, ensuring new information was integrated without completely discarding previous knowledge. The discount factor enabled the bot to account for future rewards, crucial for maintaining long-term stability. Additionally, the epsilon decay strategy allowed for sufficient exploration initially while promoting exploitation as the agent learned optimal actions, striking an effective balance between exploration and exploitation.

### Challenges and Limitations

Discretizing the continuous state space into bins introduced some challenges, as it limited the resolution with which the agent could interpret its environment. Though this method simplified Q-learning implementation, finer state representation might enhance performance. Furthermore, training required a significant number of episodes to converge, partly due to the simplicity of the Q-learning algorithm, which can be slower than newer methods like Deep Q-Learning for more complex environments.

### Visual Analysis

A plot of mean rewards across episodes illustrates the agent's learning curve, showcasing its gradual improvement over time. Early fluctuations in rewards indicate the exploration phase, while later episodes show smoother, higher reward values as the bot converged toward optimal policy.

Overall, the project demonstrates the efficacy of Q-learning in real-time control applications, successfully balancing the cartpole by optimizing a discrete Q-table. While limitations exist, the results underscore Q-learning's potential for similar control tasks and reinforce the benefits of parameter tuning and exploration-exploitation strategies in reinforcement learning. Future work could explore advanced algorithms and continuous state representations to enhance the bot's efficiency and robustness.

## Conclusion

The Cartpole Balancing Bot successfully applied Q-learning to solve a classic control problem, demonstrating reinforcement learning's power in real-time decision-making tasks. Through discrete state-space representation, strategic hyperparameter tuning, and an ε-greedy exploration strategy, the bot learned to maintain pole balance effectively, achieving high cumulative rewards consistently. This project highlights Q-learning's capability to navigate continuous control challenges with limited initial knowledge of the environment, emphasizing its adaptability and applicability to similar tasks. While discretization constraints posed minor limitations, the results validate Q-learning as an effective method for autonomous control. Future work could leverage advanced algorithms and fine-grained state representation to enhance stability and performance further.

## Future Scope

Building on the success of Q-learning in balancing the Cartpole, future work could explore advanced reinforcement learning algorithms, such as Deep Q-Learning (DQN), to handle larger and more complex state spaces. Integrating neural networks would allow the agent to approximate Q-values for continuous states, enhancing precision and scalability. Additionally, techniques like reward shaping or adaptive exploration strategies could improve learning speed and stability. Beyond the Cartpole environment, this approach could be applied to real-world control tasks, such as robotic arm balancing or autonomous vehicle navigation, where precise, continuous control is essential.

## References

[1] Shyalika, C. (2019, November 16). A Beginners Guide to Q-Learning. Retrieved September 14, 2020, from https://towardsdatascience.com/a-beginners-guide-to-q-learning-c3e2a30a653c

[2] Choudhary, A. (n.d.). [DQN Formula]. Retrieved September 15, 2020, from https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/

[3] Wikipedia contributors. (2020, October 20). Q-learning. In *Wikipedia, The Free Encyclopedia*. Retrieved 00:03, October 29, 2020, from https://en.wikipedia.org/w/index.php?title=Q-learning&oldid=984486286