Report on Applying and Comparing Q-learning and Proximal Policy Optimization (PPO) for Balancing a Pole

Introduction

The task involves implementing and comparing two different reinforcement learning (RL) algorithms—Q-learning and Proximal Policy Optimization (PPO)—to solve the CartPole-v1 problem, a classic control task. The objective is to balance a pole on a cart by applying forces to the cart. The agent's performance will be evaluated based on the learning curves, average reward per episode, and the stability of the learning process.

Reinforcement Learning Algorithms

1. **Q-learning:**
    - Q-learning is a model-free, value-based RL algorithm that aims to learn the optimal action-selection policy using a Q-table. The Q-table stores the expected utility (Q-value) of taking an action in a given state.
    - The agent updates its Q-values iteratively using the Bellman equation.
    - For this task, the state space was discretized into bins to manage the continuous state space of the CartPole environment.
2. **Proximal Policy Optimization (PPO):**
    - PPO is a policy-based RL algorithm that improves upon previous policy gradient methods by using a clipped objective to avoid large policy updates, which can destabilize training.

Implementation Details

Q-learning Implementation

- **Environment:** CartPole-v1 from OpenAI's Gym library.
- **State Space:** Discretized into bins for position, velocity, angle, and angular velocity.
- **Action Space:** Two possible actions (move cart left or right).
- **Q-table:** Initialized using `defaultdict`, where each state-action pair is associated with an initial Q-value of zero.
- **Parameters:**
    - Learning rate (α\alphaα): 0.1
    - Discount factor (γ\gammaγ): 0.99
    - Exploration rate (ϵ\epsilonϵ): 0.1
    - Number of episodes: 500

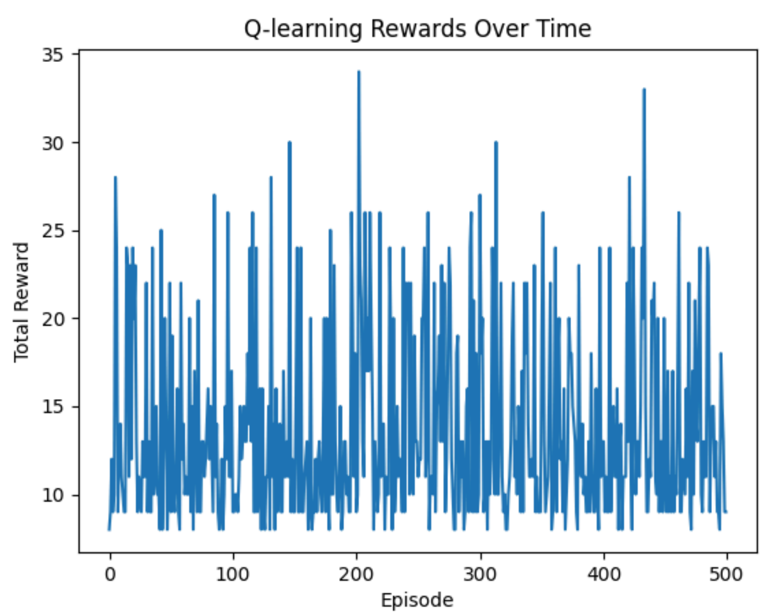
The algorithm was trained for 500 episodes, and the cumulative reward for each episode was recorded.

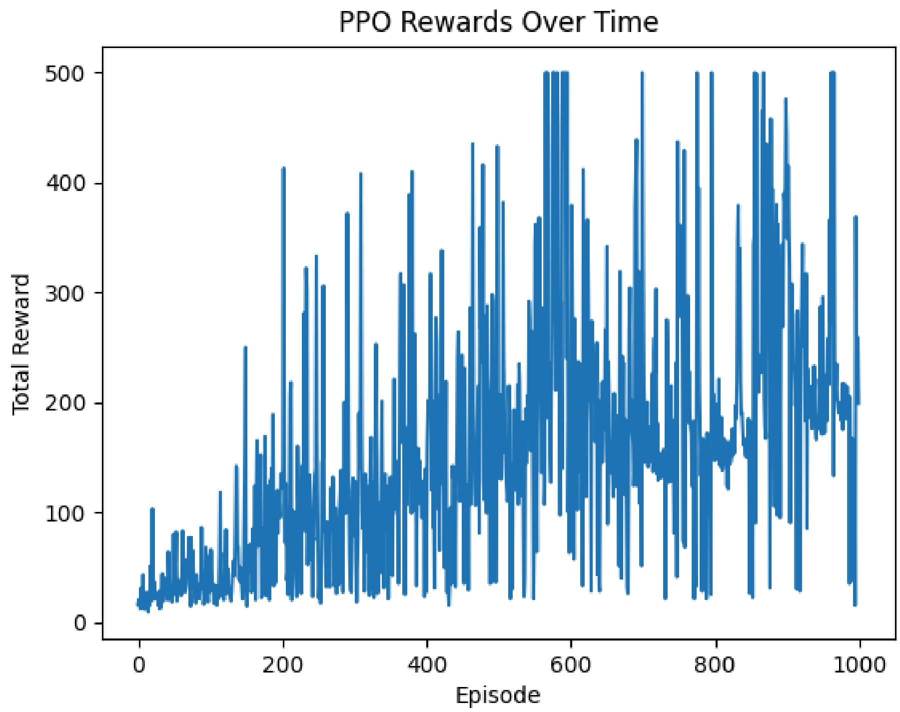
PPO Implementation

- **Environment:** CartPole-v1 from OpenAI's Gym library.
- **State Space:** Continuous, represented by the raw observations (position, velocity, angle, angular velocity).
- **Action Space:** Two possible actions (move cart left or right).
- **Policy Network:** A neural network with two hidden layers (128 neurons each) and separate heads for action selection and value prediction.
- **Optimizer:** Adam optimizer with a learning rate of 0.001.
- **Parameters:**
    - Discount factor (γ): 0.99
    - Clipping epsilon (ϵ\epsilon): 0.2
    - Number of epochs per update (KKK): 4
    - Horizon length: 2000 steps
    - Batch size: 64
    - Number of episodes: 1000

The algorithm was trained for 1000 episodes, and the cumulative reward for each episode was recorded.

Results

1. **Q-learning:**
    - The learning curve showed that the agent initially struggled to balance the pole, with low rewards in the first few episodes.
    - Gradual improvement was observed, with the agent consistently achieving higher rewards as training progressed.
    - The average reward plateaued towards the end, indicating the agent had learned a near-optimal policy.
    - 

2. **PPO:**
    - The learning curve showed a smoother and faster increase in rewards compared to Q-learning.
    - The agent quickly learned to balance the pole, reaching near-optimal performance in fewer episodes.
    - The rewards were more stable, with less variability, indicating more consistent learning.
    - 

Discussion

- **Strengths of Q-learning:**
    - Simple to implement and understand.
    - Effective in discrete state-action spaces.
    - Well-suited for environments where the state space can be discretized without loss of critical information.

- **Weaknesses of Q-learning:**
    - Performance is highly sensitive to the choice of discretization, learning rate, and exploration strategy.
    - Scaling to environments with large or continuous state spaces is challenging.
    - Learning can be slow and unstable, particularly in complex environments.

- **Strengths of PPO:**
    - Efficient in handling continuous state and action spaces.
    - Stable and reliable performance due to clipped objective and policy gradient methods.
    - Capable of learning complex policies quickly, especially when using function approximators like neural networks.
- **Weaknesses of PPO:**
    - More complex to implement compared to Q-learning.
    - Requires tuning multiple hyperparameters, such as the learning rate, clipping epsilon, and batch size.
    - Computationally more expensive due to the need for training a neural network.

Conclusion

Both Q-learning and PPO successfully learned to balance the pole in the CartPole-v1 environment, but PPO demonstrated superior performance in terms of speed, stability, and handling of continuous state spaces. For more complex problems, PPO would be a more suitable choice due to its robustness and scalability. However, for simpler or discrete environments, Q-learning remains a viable and straightforward option.