Jose Cortes December 2, 2024

Reinforcement Learning with Q–Learning on Taxi–v3 Environment

Introduction

Reinforcement learning (RL) is a powerful approach for training agents to make decisions in complex environments by learning from interaction and feedback. This project explores the application of RL, specifically Q-Learning, in solving the Taxi-v3 environment. The goal is to develop an agent capable of efficiently navigating the environment, picking up passengers, and dropping them off at the correct destinations while minimizing penalties.

The Taxi-v3 environment was chosen for its simplicity and structured reward system, which makes it an excellent domain for studying the fundamentals of reinforcement learning. By implementing Q-Learning, tuning hyperparameters, and analyzing the agent's performance, this project aims to demonstrate the potential and challenges of RL methods. The insights gained here lay the groundwork for exploring more advanced techniques like Approximate Q-Learning and Deep Q-Learning in future work.

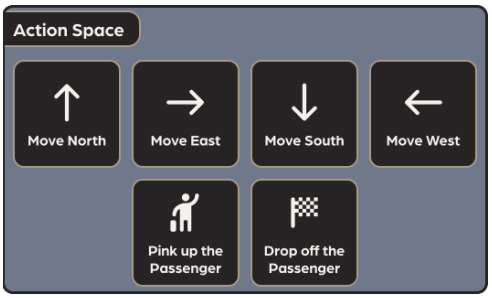
Domain Selection and Setup

For this project, I chose the **Taxi-v3 environment** as the initial domain for applying Q-Learning. Taxi-v3 is an excellent choice because it's straightforward yet effective in showcasing the fundamentals of reinforcement learning. The clear state and action space make it an ideal starting point for understanding how an agent learns through Q-Learning.

Why Taxi–v3?

Taxi-v3 demonstrates Q-Learning's strength in solving discrete-state problems. Its simplicity allows us to observe how Q-values evolve over time as the agent explores and improves its policy. This makes it a great learning tool and a baseline for more complex environments.

State and Action Space



The Taxi-v3 environment has a finite state space of 500 states, created by the combination of taxi positions, passenger locations, and destinations. These actions are deterministic, meaning their outcomes are predictable based on the current state.

Reward Structure

The environment has a well-defined reward structure:

- **Positive reward**: Successfully dropping off the passenger.
- **Penalties**: For illegal actions like dropping off the passenger at the wrong location.
- **Step penalty**: A small negative reward for each step to encourage efficiency.

The sparse rewards add complexity, as the agent must take multiple steps to achieve a positive reward. Meanwhile, penalties for incorrect actions push the agent to learn a precise and efficient policy.

The Agent's Goal

The goal for the agent is straightforward:

- Pick up the passenger.
- Drop them off at the correct destination.
- Achieve this in as few steps as possible while avoiding penalties.

This environment effectively showcases the capabilities of Q-Learning and serves as a foundation for experimenting with more advanced reinforcement learning methods.

Q–Learning: Insights and Key Design Elements

Reinforcement learning thrives on structure and adaptability. Here, we dive into the specifics of our Q-Learning implementation for the Taxi-v3 environment, outlining its core design, the agent's interaction dynamics, and how its learning process evolved.

Q–Table Design and Initialization

At the heart of Q-Learning is the Q-table, a two-dimensional array mapping states to actions. Each cell in this table represents the expected reward for a specific state-action pair. Initially, all values in the table are set to 0, reflecting the agent's lack of knowledge about the environment. Over time, this table gets populated with values as the agent learns through interaction.

Updating Q–Values with the Bellman Equation

Learning occurs through the Bellman equation, which updates the Q-values based on the agent's experiences:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r \ + \gamma \ max_a \ Q(s',a') - \ Q(s,a)\right)$$

Here:

- $\boldsymbol{\alpha}$ is the learning rate, determining the weight of new information.
- $\boldsymbol{r}$ is the reward for the current action.
- $\boldsymbol{\gamma}$ ($discount\ factor$) balances the importance of immediate versus long-term rewards.

This iterative update process refines the agent's understanding of the best actions to take in different states.

Balancing Exploration and Exploitation

To decide between exploration (trying new actions) and exploitation (choosing known good actions), the agent employs an ϵ-greedy strategy. It explores with probability ϵ, gradually decreasing over time, and exploits the learned policy otherwise. This balance ensures the agent discovers new strategies while focusing on maximizing rewards as training progresses.

Defining the End of an Episode

Each episode concludes when the agent successfully delivers the passenger to the correct destination or when a predefined maximum number of steps is reached. This episodic structure provides clear boundaries for learning cycles and allows the agent to reset and try new strategies.

Agent–Environment Interaction and Feedback

The agent learns by interacting with the environment: performing actions, transitioning to new states, and receiving feedback in the form of rewards or penalties. Positive rewards encourage actions that bring the agent closer to its goal, while penalties discourage inefficiency or errors, like picking up or dropping off passengers incorrectly.

Overcoming Challenges in Sparse Reward Scenarios

One notable challenge in Taxi-v3 is the sparse reward system. The agent must execute multiple correct steps before receiving a positive reward, which initially makes learning slow. Additionally, penalties for incorrect actions add complexity but guide the agent toward more precise behavior. By persistently updating the Q-values, the agent overcomes these hurdles and develops an efficient policy.

Training Evolution and Behavior

At the start of training, the agent explores randomly and often makes mistakes. As training progresses and the Q-values improve, the agent begins to favor actions that maximize rewards. Over time, it becomes more goal-oriented, efficiently navigating the environment and reducing unnecessary steps or penalties.

Opportunities for Advanced Enhancements

Looking forward, transitioning to Approximate or Deep Q-Learning could unlock new possibilities for more complex environments:

- Replace the Q-table with a neural network to approximate Q-values.
- Enable the agent to handle continuous or high-dimensional state spaces.
- Experiment with alternative exploration strategies like Boltzmann exploration or adaptive epsilon decay.

By incorporating these techniques, the agent's performance can be further enhanced, paving the way for tackling more sophisticated problems.

Hyperparameter Tuning: Finding the Right Balance

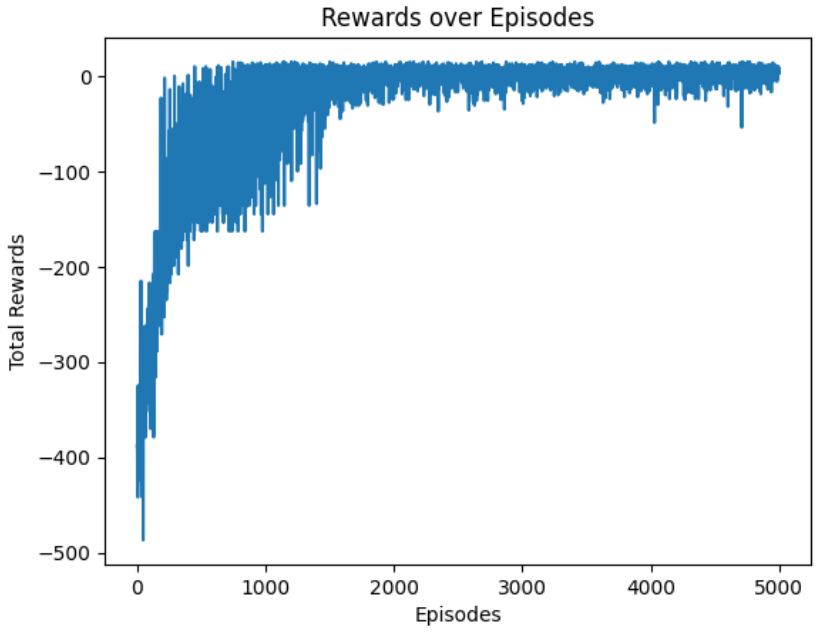
Hyperparameter tuning is a crucial part of optimizing a reinforcement learning model. For this project, the focus is on systematically testing key parameters to understand their impact on the Taxi-v3 agent's performance. The tuning process is designed to balance exploration, learning stability, and long-term rewards.

Learning Rate (α)

Starting with a learning rate of 0.1 provides a balanced approach, allowing updates to the Q-values without large fluctuations. Both constant and decaying learning rates will be tested:

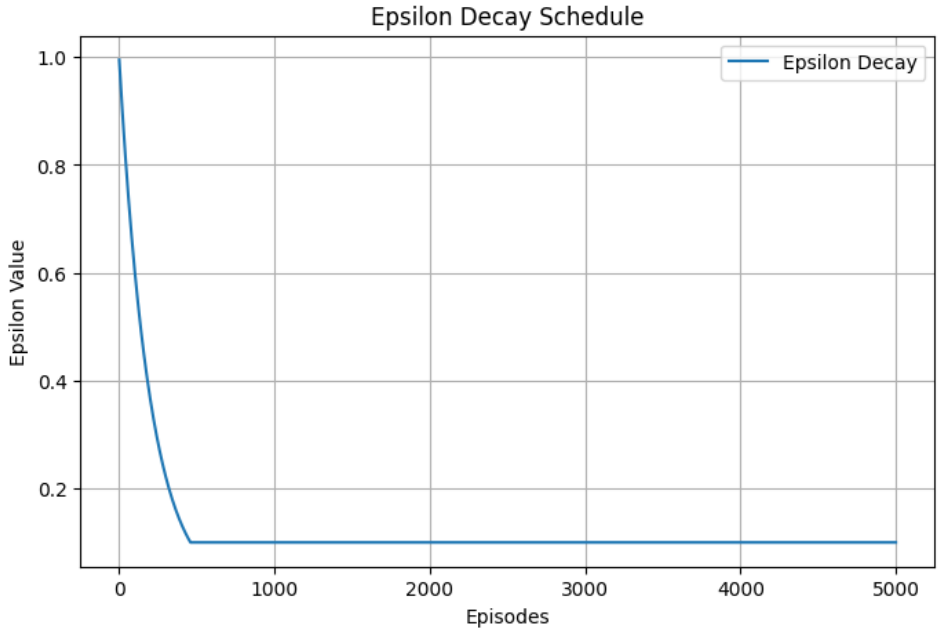
- Constant rate: Maintains a steady update step throughout training.
- Decaying rate: Gradually decreases, using a simple decay function like dividing by the number of episodes.

The impact of the learning rate will be assessed through cumulative rewards and convergence speed, with graphs illustrating how quickly the agent learns under each configuration.



Discount Factor (γ)

To encourage the agent to prioritize long-term rewards, an initial gamma value of 0.9 is selected. Additional values such as 0.7 and 0.99 will be tested to explore the trade-offs between immediate and future rewards. The cumulative rewards will highlight which gamma value results in the most effective learning policy.



Exploration Strategy $\left((\epsilon) - \boldsymbol{Greedy}\right)$

Exploration begins with epsilon set to 1 for full exploration, gradually reducing over time:

- Exponential decay: The default strategy, reducing epsilon at a consistent rate.
- Linear decay: An alternative approach if exponential decay proves ineffective.

A fixed epsilon vs. decaying epsilon strategy will also be compared, with learning curves tracking the agent's performance across these configurations.

Performance Metrics and Training Configuration

Performance is evaluated using:

- Cumulative rewards over episodes: The primary metric to measure learning progress.
- Convergence speed: The number of episodes needed to achieve stable high rewards.

Training will run for 1000 episodes per configuration, striking a balance between thorough testing and computational feasibility. Learning curves (reward vs. episodes) will visualize the results, highlighting the impact of different hyperparameter combinations.

Systematic Experimentation

Rather than tweaking one parameter at a time, configurations will be tested in pairs (e.g., combining different learning rates with epsilon decay strategies). This systematic approach will provide deeper insights into how hyperparameters interact and influence the agent’s learning process.

By experimenting with these variables, the agent’s performance can be optimized, setting a solid foundation for tackling more complex environments in the future.

Training Performance and Analysis

Cumulative Rewards

During training, the agent demonstrated clear improvement in earning cumulative rewards. Initially, rewards were sparse due to the agent's exploratory actions and lack of knowledge about the environment. Over time, as the Q-values were updated and the agent learned an effective policy, the cumulative rewards increased steadily. This upward trend reflects the agent's growing capability to achieve positive outcomes consistently.

Occasionally, the rewards plateaued, which likely occurred when the agent required further exploration to discover better strategies or adapt to sparse reward signals in the

environment. These periods were temporary, as exploration strategies like epsilon decay ensured the agent continued to refine its policy.

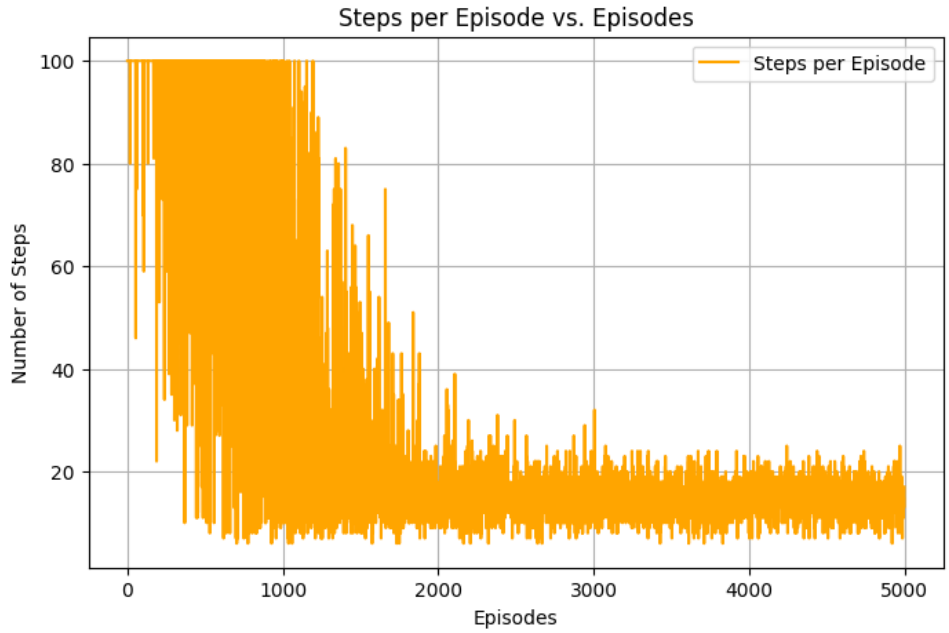
Steps Per Episode

The number of steps the agent needed to complete an episode decreased significantly over the training period. Early on, the agent often took close to the maximum allowable steps, as it was still learning the most effective actions to take in each state. However, by the later episodes, the agent consistently completed the task in fewer steps, nearing optimal efficiency.

This decrease in steps indicates that the agent not only learned to achieve the task but also improved its efficiency in doing so. Each episode provided feedback that guided the agent toward refining its behavior, minimizing penalties, and optimizing rewards.

Performance Consistency

After approximately 300–400 episodes, the agent reached a point of consistent performance, reliably completing episodes with near-optimal actions. While there were occasional drops in performance due to the agent exploring new actions, these instances were rare and quickly corrected as the Q-values guided the agent back toward the optimal policy.



Visual Representations

The learning curve, which plots cumulative rewards against episodes, showed a gradual upward trajectory with minor fluctuations early on. These fluctuations correspond to the agent's

exploration phase. As training progressed, the curve leveled off, signifying the agent's convergence to a stable and effective policy.

Similarly, a plot of steps per episode revealed a clear downward slope. Initially, the agent's inefficiency was evident, but this trend improved sharply in the early stages of training. The slope became more gradual as the agent neared its optimal policy, reflecting a steady refinement process.

Overall Evaluation

The Q-Learning agent successfully achieved the goals of the Taxi-v3 environment. It learned to navigate the grid, pick up passengers, and drop them off at their destinations with minimal penalties. The agent's training evolution—from random exploration to near-optimal performance—underscores the effectiveness of Q-Learning for discrete-state problems like Taxi-v3.

Despite initial challenges with sparse rewards and penalties for incorrect actions, the agent adapted well, leveraging the structured reward system to guide its learning. These results validate the choice of Q-Learning for this task and provide a strong foundation for advancing to more complex reinforcement learning methods in future iterations of this project.

Novel Contributions and Future Work

This project highlights the adaptability of Q-Learning in solving the Taxi-v3 environment, offering a structured approach to reinforcement learning for discrete-state problems. One notable contribution is the detailed analysis of hyperparameter effects on learning efficiency, which provides a clear understanding of how exploration and learning rates influence agent behavior. This methodical experimentation forms a practical guide for tuning similar models in other domains.

Future work could involve extending this approach to more complex environments using Approximate or Deep Q-Learning. Incorporating neural networks would enable the agent to handle continuous state spaces and higher-dimensional problems. Additionally, exploring alternative exploration strategies, such as adaptive epsilon decay or curiosity-driven learning, might further enhance the agent's efficiency and adaptability. These advancements could expand the applicability of this work to dynamic and real-world scenarios.