# CartPole Q-Learning with Gridded State Space

## 1. Introduction

### 1.1 The CartPole Problem

This report details an experiment applying the Q-learning reinforcement learning algorithm to the classic CartPole control problem, specifically the CartPole-v1 environment provided by the Gymnasium library. The objective in CartPole is to prevent a pole, hinged to a movable cart, from falling over. The agent controls the cart by applying discrete forces (left or right) at each time step. The episode ends when the pole angle exceeds a certain threshold, the cart moves beyond the defined world boundaries, or a maximum number of steps (often 500 in standard implementations, though not explicitly limited in this training run) is reached.

### 1.2 Challenge: Continuous State Space

The main difficulty when applying traditional Q-learning methods to CartPole stems from its continuous state environment. The state of the environment consists of four continuously changing variables including cart position and velocity as well as pole angle and angular velocity.Tabular methods require a finite set of discrete states.

### 1.3 Approach: State Space Discretization

To overcome this challenge, this implementation employs state space discretization, often referred to as "gridding" or "tiling." The continuous four-dimensional state space is partitioned into a finite grid of hypercubes (bins). Each bin represents a discrete state, allowing the use of a tabular Q-table to store and update state-action values. The goal of this experiment is to train an agent using Q-learning on this discretized state space and evaluate its ability to balance the pole.

## 2. Methodology

### 2.1 Q-Learning Algorithm

Q-learning is an off-policy, model-free reinforcement learning algorithm.

* **Model-Free:** It learns the optimal policy directly from interactions with the environment without needing to build a model of the environment's dynamics.
* **Off-Policy:** It learns the value of the optimal policy independently of the agent's exploration policy (in this case, epsilon-greedy).
* **Q-Table:** It learns a state-action value function, Q(s, a), which represents the expected cumulative discounted future reward for taking action 'a' in state 's' and following the optimal policy thereafter. The Q-values are stored in a table where rows/indices correspond to discrete states and columns correspond to actions (left/right).
* Update Rule: Q-values are iteratively updated using the Bellman equation:  
  Q(s, a) <- Q(s, a) + alpha \* [r + gamma \* max\_a'(Q(s', a')) - Q(s, a)]  
  where s is the current state, a is the action taken, r is the reward received, s' is the next state, alpha is the learning rate, and gamma is the discount factor.

### 2.2 State Space Discretization Details

The continuous state space was mapped to a discrete grid:

* **Number of Bins (nbins):** Each of the four state dimensions was divided into 20 bins (nbins = 20). This results in a total of 20^4 = 160,000 discrete states.
* **State Boundaries:** Specific boundaries were defined for discretization:
  + Cart Position: [-0.4, 0.4] (a = 0.4)
  + Cart Velocity: [-2.0, 2.0] (b = 2.0)
  + Pole Angle: [-0.3, 0.3] radians (c = 0.3)
  + Pole Angular Velocity: [-0.1, 0.1] radians/sec (d = 0.1)  
    Observations outside these ranges were likely clipped or assigned to the outermost bins. The choice of these boundaries and the number of bins significantly impacts performance; too coarse a grid might merge distinct situations (state aliasing), while too fine a grid increases the state space size dramatically, slowing learning.
* **Q-Table Initialization:** A Q-table of size (160000 states x 2 actions) was initialized, likely with zeros or small random values, before training commenced.

### 2.3 Training Hyperparameters

The learning process was guided by several key hyperparameters:

* **Number of Episodes (n\_episodes = 100,000):** A substantial number of episodes were used to allow the agent sufficient experience to explore the state space and converge towards optimal Q-values.
* **Discount Factor (gamma = 0.9):** This value determines the importance of future rewards. A value of 0.9 means rewards received one step in the future are worth 90% of immediate rewards, two steps away are worth 81%, and so on. It encourages the agent to find long-term stability rather than just immediate survival.
* **Exploration Probability (epsilon = 0.2):** Using an epsilon-greedy strategy, the agent chose a random action with 20% probability and the action with the highest Q-value (greedy action) with 80% probability. This balance ensures the agent explores potentially better actions while exploiting its current knowledge. A constant epsilon was used here, though decaying epsilon schedules are common.
* **Learning Rate (alpha = 0.1):** This parameter controls the step size for Q-value updates. A value of 0.1 means that the new Q-value estimate is a weighted average, with 10% weight given to the new information (target value) and 90% to the old Q-value. It dictates how quickly the agent adapts to new information.
* **Logging Interval (log\_interval = 5000):** Provided periodic feedback on training progress.

## 3. Training Process and Diagnostics

### 3.1 Training Loop Execution

The core training loop iterated 100,000 times (episodes). Within each episode:

1. The CartPole environment was reset to an initial state.
2. The continuous state observation was mapped to its corresponding discrete grid cell (state s).
3. An action a was selected using the epsilon-greedy policy based on the Q-values for state s.
4. The action was executed in the environment, yielding a reward r (typically +1 for each step survived) and the next continuous observation.
5. The next continuous observation was mapped to the next discrete state s'.
6. The Q-value Q(s, a) was updated using the Q-learning rule, incorporating the reward r and the maximum Q-value available from the next state s'.
7. The state transitioned (s <- s').
8. Steps 3-7 repeated until the environment signaled termination (pole fell, cart out of bounds, max steps).

### 3.2 Monitored Training Metrics

Several metrics provided insights into the learning dynamics:

* **Game Length:** Directly measured the agent's performance within each training episode. An increasing trend indicates learning.
* **Q-Value Change Norms (L2 and Infinity):** These norms measure the magnitude of changes to the entire Q-table during an episode.
  + *Infinity Norm (inf\_norm):* The maximum absolute change in any single Q-value entry.
  + L2 Norm (l2\_norm): The square root of the sum of squared changes across all Q-value entries.  
    Decreasing trends in these norms generally suggest that the Q-values are converging towards stable estimates.
* **Visited States:** Tracking state visitations helps understand the extent of state space exploration.
* **Terminal States:** Identifying states where episodes frequently terminate can highlight challenging regions of the state space.

### 3.3 Analysis of Training Observations

* **Training Duration:** The entire training process completed in approximately 45.87 seconds, indicating relatively fast computation per episode despite the large number of episodes.
* **Learning Trend:** The scatter plot of game length versus episode number showed considerable noise but a discernible upward trend, particularly in the maximum lengths achieved in later episodes. This confirms that the agent progressively learned to balance the pole for longer durations. The periodic log output corroborated this, showing increases in episode length over time (e.g., from 12 at episode 0 to peaks over 100 later on).
* **Convergence Behavior:** The plots for L2 and infinity norms exhibited a general decrease over the training period. This suggests that the Q-values were stabilizing as the agent gained more experience. However, the persistence of non-zero updates indicates that learning was likely still ongoing, or that the stochasticity of the environment and exploration kept Q-values fluctuating.
* **Training Performance Variability:** The histogram of training game lengths showed a wide spread, heavily influenced by the epsilon = 0.2 exploration. Random actions often lead to premature termination, masking the agent's true capability during training but being essential for discovering the optimal policy.

## 4. Evaluation of the Learned Policy

### 4.1 Evaluation Protocol

Following training, the agent's learned knowledge was evaluated deterministically.

* **Policy Derivation:** A greedy policy (pi) was derived from the final Q-table. For any given states, this policy selects the action a that maximizes Q(s, a).
* **Testing:** The agent executed this greedy policy over n\_test\_episodes = 10,000 independent episodes. Exploration (epsilon) was set to 0 during this phase to assess the pure exploitation capability of the learned Q-values. The length (number of steps survived) of each test episode was recorded.

### 4.2 Performance Analysis

The distribution of episode lengths during testing provides a measure of the final policy's effectiveness and consistency:

* **Central Tendency:** The mean episode length was 72.83 steps, and the median was 65 steps. This indicates a typical performance level achieved by the agent.
* **Variability:** The standard deviation was quite high (37.95 steps), suggesting significant inconsistency in performance across different episodes. This variability is also evident in the range between the minimum (16 steps) and maximum (1148 steps) and the interquartile range (IQR) of 29 steps (82 - 53).
* **Distribution Shape:** The histogram and the high maximum value relative to the mean/median clearly show a right-skewed distribution. While the agent performed moderately well on average, it occasionally achieved very long balancing times (outliers like 1148 steps). This suggests the policy is effective in certain state trajectories but may lack robustness or fail quickly in others. The discretization might contribute to this, where crucial state differences are potentially masked within the same grid cell.
* **Comparison to Standard:** While CartPole-v1 is often considered "solved" if an agent consistently achieves an average reward of 195 over 100 episodes (or 475 for newer versions), this implementation's average of ~73 steps falls short of that benchmark. This is not unexpected for a basic tabular Q-learning approach with a relatively coarse grid and without hyperparameter tuning.

## 5. Conclusion

### 5.1 Summary of Findings

This experiment successfully implemented Q-learning for the CartPole problem by discretizing its continuous state space into a 20x20x20x20 grid. Training over 100,000 episodes with an epsilon-greedy strategy demonstrated clear learning, evidenced by increasing episode lengths and converging Q-value norms. The resulting greedy policy achieved an average episode length of approximately 73 steps over 10,000 test episodes, with considerable variability but occasional high performance.

### 5.2 Limitations

* **State Space Discretization:** The gridding approach is inherently limited. The choice of bin boundaries and the number of bins (nbins) is crucial and often requires tuning. It suffers from the "curse of dimensionality" – the number of states grows exponentially with the number of dimensions and bins, making it infeasible for higher-dimensional problems. Furthermore, discretization introduces approximation errors, potentially merging critically different states into the same bin (state aliasing).
* **Constant Epsilon:** Using a constant exploration rate (epsilon = 0.2) throughout training might be suboptimal. A decaying epsilon schedule often leads to better final performance by prioritizing exploration early and exploitation later.
* **Hyperparameter Sensitivity:** The performance is sensitive to the choice of alpha, gamma, epsilon, and the grid parameters. The values used (0.1, 0.9, 0.2, 20 bins) were not necessarily optimized.

### 5.3 Potential Improvements and Future Work

* **Hyperparameter Tuning:** Systematically tune alpha, gamma, epsilon (potentially with a decay schedule), and the grid parameters (nbins, boundaries) to improve performance.
* **Adaptive Discretization:** Use more sophisticated discretization techniques where the grid resolution is adapted based on experience (e.g., finer grid in critical regions).
* **Function Approximation:** Replace the tabular Q-table with a function approximator (e.g., linear function, neural network) to handle the continuous state space directly. This leads to methods like Deep Q-Networks (DQN), which can often achieve better performance and scale to more complex problems.
* **Alternative Algorithms:** Explore other RL algorithms like SARSA, Actor-Critic methods, or policy gradient methods.

### 5.4 Final Conclusion

Despite its limitations, this experiment effectively illustrates the core principles of applying tabular Q-learning to a continuous control problem via state space discretization. It successfully trained an agent to perform the CartPole task, achieving respectable, albeit variable, performance. The results highlight both the potential and the challenges of using simple grid-based methods for continuous domains.

## 6. Questions

Q1: Why is state discretization necessary for tabular Q-learning in CartPole?

A1: Tabular Q-learning requires a finite number of discrete states to store Q-values in a table. The CartPole environment naturally has continuous state variables (position, velocity, angle, angular velocity). Discretization (gridding) converts these continuous values into a finite set of bins, making the problem solvable with a Q-table.

Q2: What is the "curse of dimensionality" in the context of this experiment?

A2: The curse of dimensionality refers to the exponential increase in the number of states (and thus the size of the Q-table and the data needed) as the number of state variables (dimensions) or the number of bins per dimension increases. In this case, with 4 dimensions and 20 bins each, we have 204=160,000 states. Adding more dimensions or bins makes the problem computationally much harder.

Q3: How do the hyperparameters (alpha, gamma, epsilon) affect learning?

A3:

* **alpha (Learning Rate):** Controls how much new Q-value estimates overwrite old ones. High alpha means faster learning but potential instability; low alpha means slower but potentially more stable learning.
* **gamma (Discount Factor):** Determines the importance of future rewards. High gamma (near 1) encourages long-term planning; low gamma (near 0) focuses on immediate rewards.
* **epsilon (Exploration Rate):** Balances exploration (trying random actions) and exploitation (using known best actions). High epsilon encourages finding new strategies but can hurt performance; low epsilon exploits current knowledge but might miss better strategies.

Q4: Why did the performance evaluation show high variability (large standard deviation and range)?

A4: Several factors contribute:

* **Discretization:** The grid might not capture the nuances of the continuous state space perfectly. Slightly different continuous states falling into the same bin might require different optimal actions, leading to suboptimal choices.
* **Policy Limitations:** The learned policy might be optimal for the *discretized* space but not perfectly robust in the *continuous* environment. Certain initial states or trajectories might lead it into situations it handles poorly.
* **Stochasticity:** Although the evaluation uses a deterministic policy, the environment itself might have inherent (though minimal in CartPole) or perceived stochasticity due to the discretization.

Q5: Could this approach work for more complex problems?

A5: Simple gridding scales poorly to problems with more state dimensions due to the curse of dimensionality. For more complex problems (e.g., robotics with many joints, game AI with large state spaces), methods using function approximation (like Deep Q-Networks) that learn features directly from high-dimensional or continuous states are generally required.

## 7. References

1. **Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press.** (The standard textbook for reinforcement learning, covering Q-learning and related concepts).
2. **Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. PhD thesis, University of Cambridge.** (Original work introducing Q-learning).
3. **Gymnasium Documentation (formerly OpenAI Gym):** <https://gymnasium.farama.org/> (Source for the CartPole-v1 environment and standard RL environments).
4. **Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym. *arXiv preprint arXiv:1606.01540*.** (Paper introducing the original OpenAI Gym framework).
5. **Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529-533.** (Introduced Deep Q-Networks (DQN), a key method for handling continuous/high-dimensional states, relevant for comparison/improvement).