Solving a Stochastic Gridworld Problem using Q-Learning

Varun Karthik, Jeevan Hebbal Manujanath, Yeshwanth Reddy Gurureddy

October 5, 2025

Contents

Abstract

This report details the application of the Q-learning algorithm to solve a stochastic 3x4 Gridworld problem. The objective is to find an optimal policy that maximizes the cumulative reward. We explore the fundamental concepts of Q-learning, analyze the impact of key hyperparameters on the agent's performance, and discuss the convergence properties of the Q-values versus the learned policy. The analysis is supported by empirical results generated from a Python implementation of the Q-learning agent.

1 Introduction to Q-Learning

Reinforcement Learning (RL) is a paradigm of machine learning where an agent learns to make decisions by interacting with an environment. The agent performs actions, receives rewards (or penalties), and observes new states, learning a policy that maximizes its long-term reward.

Q-learning is a model-free, off-policy RL algorithm. "Model-free" means it does not need a model of the environment's dynamics (i.e., it doesn't need to know the transition probabilities). It learns the value of actions directly from experience. Its primary goal is to learn an action-value function, denoted as $Q(s, a)$, which represents the expected cumulative discounted reward of taking action a in state s and following the optimal policy thereafter.

1.1 The Gridworld Environment

The problem is set in a 3x4 grid. The agent's goal is to navigate from a 'START' state at (1,1) to one of two terminal states: a goal state at (4,3) with a reward of $+1$ and a penalty state at (4,2) with a reward of -1. All other movements incur a small negative reward of -0.04 to encourage efficiency. The environment is stochastic: an intended action is successful only 80% of the time. There is a 10% chance of moving to the left of the intended direction and a 10% chance of moving to the right.

2 Methodology

2.1 The Q-Function and Update Rule

The core of the algorithm is the iterative updating of the Q-function based on the agent's experiences. This update is derived from the Bellman equation and is performed using the following rule:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

Where:

- s_t and a_t are the current state and action.
- R_{t+1} is the reward received after performing action a_t.
- s_{t+1} is the next state.
- α is the **learning rate**, controlling how much new information overrides old information.
- γ is the **discount factor**, determining the importance of future rewards.
- $\max_a Q(s_{t+1}, a)$ is the agent's estimate of the best possible future value from state s_{t+1}.

2.2 Exploration vs. Exploitation

To discover the optimal policy, the agent must balance exploring new actions with exploiting actions that have proven to be effective. We use an ϵ-greedy strategy:

- With probability ϵ, the agent chooses a random action (exploration).

- With probability $1 - \epsilon$, the agent chooses the action with the highest Q-value for the current state (exploitation).

To ensure convergence, ϵ is gradually decayed over time, shifting the agent's focus from exploration to exploitation as it learns more about the environment.

3 Results and Discussion

The agent was trained for 20,000 episodes, and its performance was evaluated based on the effect of different hyperparameters.

3.1 Effect of Hyperparameters

The choice of hyperparameters significantly affects the agent's learning efficiency and final performance.

3.1.1 Learning Rate (α)

As shown in Figure 1, the learning rate dictates how quickly the agent adapts. A low α (0.01) leads to slow but stable learning, while a high α (0.9) causes instability. An optimal value of $\alpha = 0.1$ provides a balance, allowing for rapid yet stable convergence.

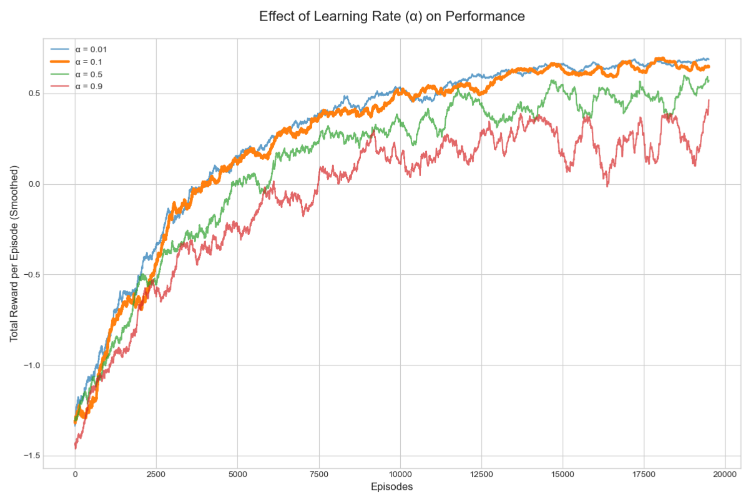

Figure 1: Effect of Learning Rate (α) on Performance. The optimal value of 0.1 shows a smooth and rapid convergence to a high reward.

3.1.2 Discount Factor (γ)

The discount factor controls the agent's foresight. Figure 2 shows that a higher γ (e.g., 0.99) is essential for the agent to value the distant +1 reward and find the optimal path. Lower values make the agent too "short-sighted."

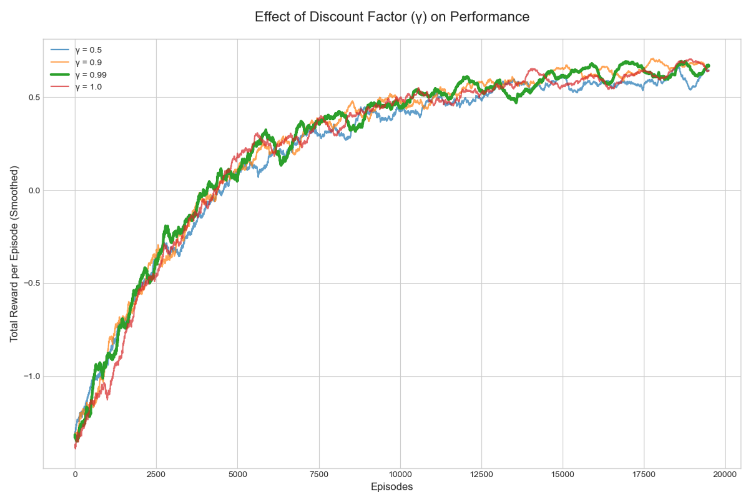

Figure 2: Effect of Discount Factor (γ) on Performance. Higher values that encourage looking far into the future lead to better overall rewards.

3.1.3 Epsilon (ϵ) Decay Schedule

The rate at which the agent reduces exploration is crucial. Figure 3 illustrates this trade-off. A fast decay (0.999) causes the agent to stop exploring too early, settling on a suboptimal policy. A slow decay (0.99999) wastes too much time on random actions. The optimal rate (0.9999) allows for sufficient exploration before committing to the best-found policy.

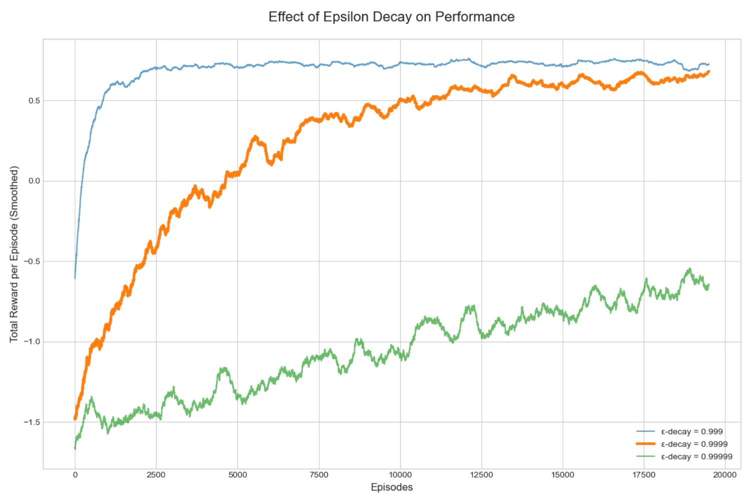

Figure 3: Effect of Epsilon (ϵ) Decay on Performance. A balanced decay rate is critical for effective learning.

3.2 Convergence of Q-Values vs. Policy

The results strongly indicate that **the policy converges before the Q-values**. An agent's policy is determined by the *relative* ranking of Q-values for actions in a given state ($\arg\max_a Q(s,a)$). This ranking can stabilize long before the Q-values themselves stop changing.

Figure 4 shows the clear, stable final policy with the standard penalty. In contrast, Figure 5 shows that the specific Q-value for taking 'Up' in the start state is still being refined late into training, even though the policy itself has likely been fixed for thousands of episodes.

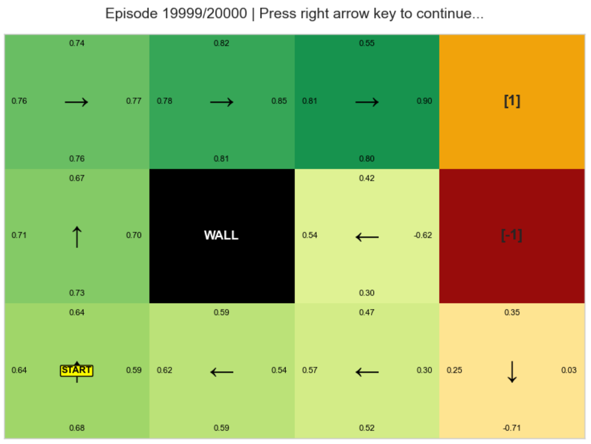

Figure 4: The final learned policy with a penalty of -1. The arrows indicate a direct, optimal path to the +1 reward.

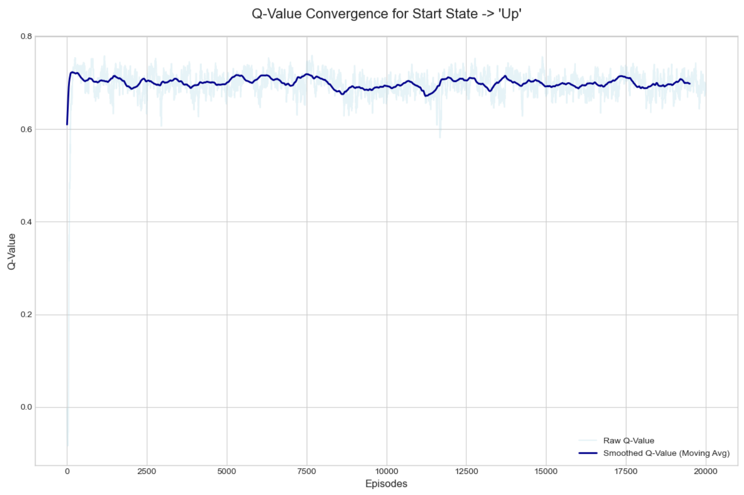

Figure 5: Convergence of a single Q-value (Start State, Action 'Up'). The value continues to fluctuate and be refined even after the policy has stabilized.

3.3 Effect of Increased Penalty

When the penalty at state (4,2) was increased from -1 to -200, the agent learned a much more conservative and risk-averse policy. A direct comparison between the two resulting policies reveals the agent's ability to factor in the magnitude of risk.

- **Standard Policy (Penalty -1):** As seen in Figure 4, the agent learns the most direct path to the +1 reward. This path travels directly adjacent to the -1 penalty state. Given the stochastic nature of the environment, there is a 10% chance of slipping into the penalty state from the cell below it (3,2). With a small penalty, this risk is acceptable for the sake of efficiency.
- **Risk-Averse Policy (Penalty -200):** Figure 6 shows a dramatically different strategy. The agent learns to take a longer, circuitous route, going up and around the wall. It completely avoids any state

from which it could accidentally slip into the high-penalty zone. The Q-values for the states near the -200 penalty are significantly lower, propagating a ”fear” of that region throughout the grid.

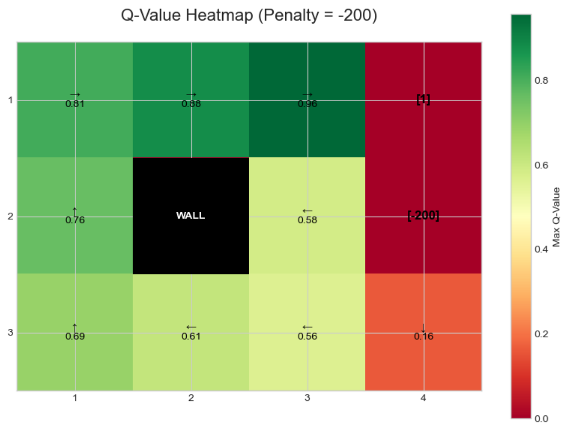

Figure 6: The final learned policy with a penalty of -200. The agent learns a longer, safer path to avoid any possibility of receiving the massive penalty.

The significance of this comparison is that it demonstrates a core strength of Q-learning. The algorithm doesn’t just learn a binary good/bad path; it learns a nuanced policy based on the *expected value* of actions. A 10% chance of a -200 reward is a massive deterrent, and the agent correctly learns that the longer, safer path has a higher overall expected reward than the shorter, riskier path. This ability to make trade-offs between efficiency and risk is crucial for applying reinforcement learning to real-world problems.

4 Conclusion

Q-learning successfully determined an optimal policy for the stochastic Gridworld. The analysis demonstrates the critical role of hyperparameter tuning in achieving efficient learning. We observed that an agent’s policy converges substantially earlier than its underlying Q-values. Furthermore, the model’s adaptation to a drastically increased penalty highlights its robustness and its ability to learn a risk-averse behavior that is appropriate to the environment’s dangers.