HW4, Markov Decision Process, Hossein Sahour, PhD

Background

The article explores two different Markov Decision Process (MDP) challenges, employing three reinforcement learning algorithms as solution: value iteration, policy iteration, and the model-free Q-learning method.

Markov decision problems

We used two interesting problems namely the Gamblres problem and the Grid world problem.

1) **Explanation of Gambler's problem**

The Gambler's Problem is a well-known example in reinforcement learning that models a situation where a gambler tries to reach a certain wealth goal by betting on the outcomes of coin flips. In this problem, the gambler begins with an initial capital and can bet any amount up to their current capital in each round. The result of each bet is binary: win or lose. This is determined by a biased coin flip with a known probability of winning. The capital amount will change by winning or losing in each bet. The final goal of the gambler is to reach a predefined amount of wealth without losing all their money. The challenge lies in devising an optimal policy that maximizes the probability of reaching the target before going bankrupt, considering the stochastic nature of the game and the current capital of the gambler.

Why Gambler's problem is interesting?

The Gambler's problem is an example of decision-making under uncertainty which reflects many real-world scenarios. It shows how to formulate and solve problems where success is probabilistic and is subjected to both current states and chosen actions (amount of bet). Additionally, this problem provides valuable insights into how strategic decisions can be optimized in environments that inherently involve risk and variability.

2) **The Grid World problem**

The Grid World problem is a simple, illustrative environment where an agent navigates through a grid of cells (states) to reach a specific location while minimizing the number of steps or maximizing cumulative rewards. Each cell in the grid represents a distinct state, and the agent can move in different directions to transition from one state to another. In this exercise, we used a simple grid size world 20x20 (400 states). The reward is -1 for each move, to find the shortest path to a goal.

Why Grid World problem is interesting?

The grid world problem is a simple yet powerful tool to show the principles of RL. It helps in understanding how agents learn to make decisions over time to achieve goals. This problem allows for a comparison between various reinforcement learning (RL) algorithms. Also, it introduces key concepts in MDP like state space, action space, reward shaping, and the notion of policy.

Methodology

We used two approaches to solve the problems. First using value iteration and policy iteration (section1). Second, we used a model-free algorithm namely Q-learning (section 2).

Value iteration

Value Iteration aims to find the optimal values for each state in an MDP. The value of a state is the total amount of reward an agent can expect to collect in the future, starting from that state. The algorithm initializes values arbitrarily for all states. It then repeatedly updates the value of each state using the Bellman Optimality Equation. This update considers the rewards received from taking all possible actions in the current state, then moving to the next state, and receiving the future rewards (discounted by a factor gamma). The process repeats until the values converge, meaning the change in values between iterations falls below a small threshold.

Policy Iteration

Policy Iteration aims to find the optimal policy in an MDP through two main steps of policy evaluation and policy improvement. For a given policy, policy evaluation calculates the value of each state by assuming that the agent follows this policy. Policy Improvement updates the policy at each state to perform the best action according to the evaluated values. Finally, the two steps are repeated until the policy stops changing, showing convergence to the optimal policy.

Q-Learning

Q-learning is a model-free RL algorithm. It learns the value of an action in a particular state, called the Q-value, without requiring a model of the environment. The algorithm maintains a Q-table, which is updated iteratively. Each Q-value shows the expected reward for choosing a certain action in a specific state and then following the best strategy after that. The Q-values are updated using the Bellman Equation, incorporating a learning rate (alpha) to balance between old and new information. The policy is determined by choosing the action with the highest Q-value in each state.

Section 1) Implementation of value iteration and policy iteration

MDP-1 Gambler`s problem

The gambler problem in our exercise has 101 discrete states, representing the gambler's capital ranging from 0 to 100. The only non-zero reward is in state 100, which is set to 1, showing the goal of reaching a capital of 100 by the gambler. In each state, the gambler can bet any money between 1 and the smaller of their current capital or the amount needed to reach 100. With a 40% chance of winning (p=0.4) a bet, the gambler's capital will either go up or down by the bet amount, depending on whether they win or lose. The discount factor is 1, meaning future rewards are valued equally as present rewards. Convergence for value iteration was defined by the maximum change in the value function (max delta) across all states becoming less than a predefined threshold. For policy iteration, convergence was defined by the policy remaining the same between iterations, which means that no further improvement could be found.

The value iteration took 11 iterations and 0.0358 seconds to converge, while the policy iteration took 8 iterations and 0.0781 to converge. Policy iteration converges faster in terms of iteration count but not in terms of computation time. This is because policy iteration consists of two steps in each iteration including the policy evaluation step and policy improvement step. During policy evaluation, the value function is estimated for a fixed policy, and during policy improvement, the policy is improved based on the current value function. These steps tend to make significant changes to the policy and value

function, leading to faster convergence. However, value iteration combines policy evaluation and improvement into one step. Although this makes it faster, it typically takes more iterations because it incrementally improves the value function. Convergence for value iteration was defined by the maximum change in the value function (max delta) across all states becoming less than a predefined threshold. For policy iteration, convergence was defined by the policy remaining the same between iterations, which means that no further improvement could be found.

Figure1, which compares the final policies of both methods, shows that the policies are very similar with some differences. This might indicate that while both methods are approaching an optimal policy, they might have converged to slightly different policies. This can happen due to differences in how the algorithms handle ties between taken actions or the precision of floating-point calculations.

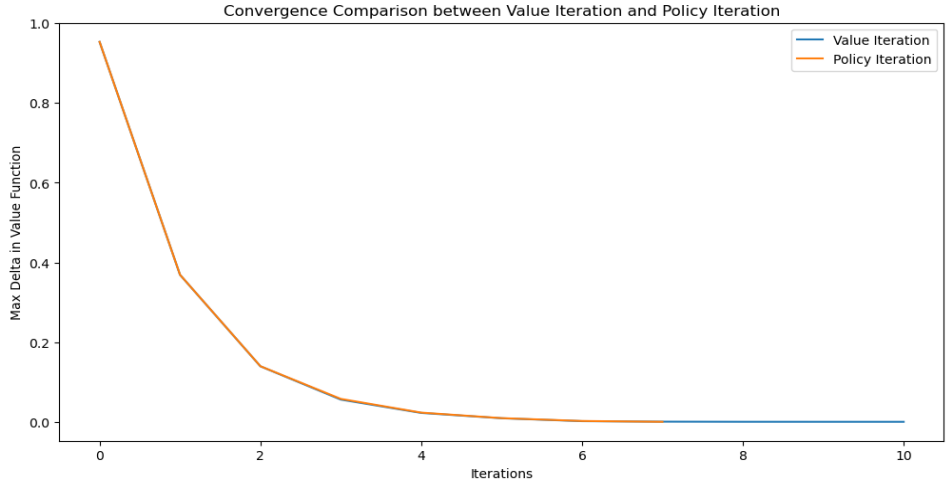

Figure 1 Convergence comparison between value iteration and policy iteration

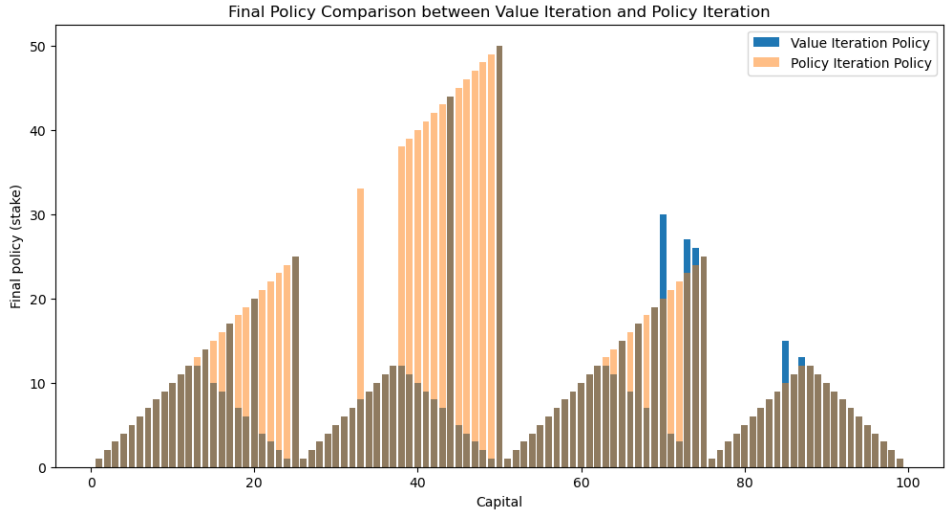

Figure 2 Final Policies obtained from value iteration and policy iteration.

Figure 2 shows the convergence of the maximum delta in the value function, suggesting that the values are converging to similar solutions. The plot shows that the changes in the value function reduce over iterations for the two methods, with policy iteration having a smoother decline.

In general, the number of states in the MDP affects the number of computations that need to be performed in each iteration. In the case of the gambler's problem, with 101 states, both value iteration and policy iteration have to process each state in every single iteration. Therefore, a higher number of states would increase computational time per iteration for both algorithms, but it doesn't necessarily affect the number of iterations required to converge.

MDP-2 Grid world problem

My grid world problem has a total of 400 states, as it's a grid world of size 20x20. There are 4 possible actions in this MDP: Up, Down, Left, and Right. The reward is -1 by default, except for transitions from states that are multiples of 5, which receive a slightly better reward of -0.5. The goal state, which is the bottom-right corner of the grid (state 399), has a reward of 0, showing completion of the process. For both value iteration and policy iteration, convergence is defined by the variable theta (set to 1e-6), which determines how small the delta should be to stop the iteration.

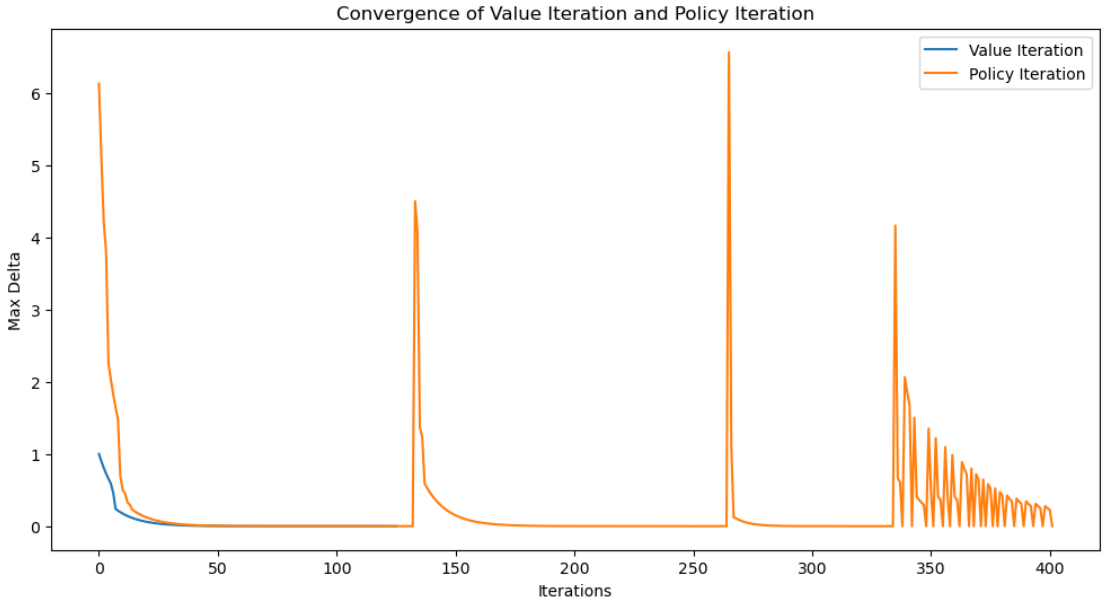

Figure 3 Convergence comparison between value iteration and policy iteration for the gambler`s problem

Solving the grid world problem, value iteration converged in 126 iterations and 33 seconds while policy Iteration converged in 26 iterations and 34 seconds. Policy Iteration converges slower in terms of computation time but requires significantly fewer iterations compared to value Iteration. Policy Iteration takes more time because it consists of two steps as we previously mentioned. This makes each iteration of policy more computationally intensive. Value Iteration, however, incrementally improves the value estimates, requiring more iterations as it indirectly improves the policy by converging the value function. Convergence is defined as the point where the maximum change in the value function or Policy Iteration, falls below the predefined threshold (theta = 1e-6). The number of states in an MDP affects the computational resources required for both algorithms.

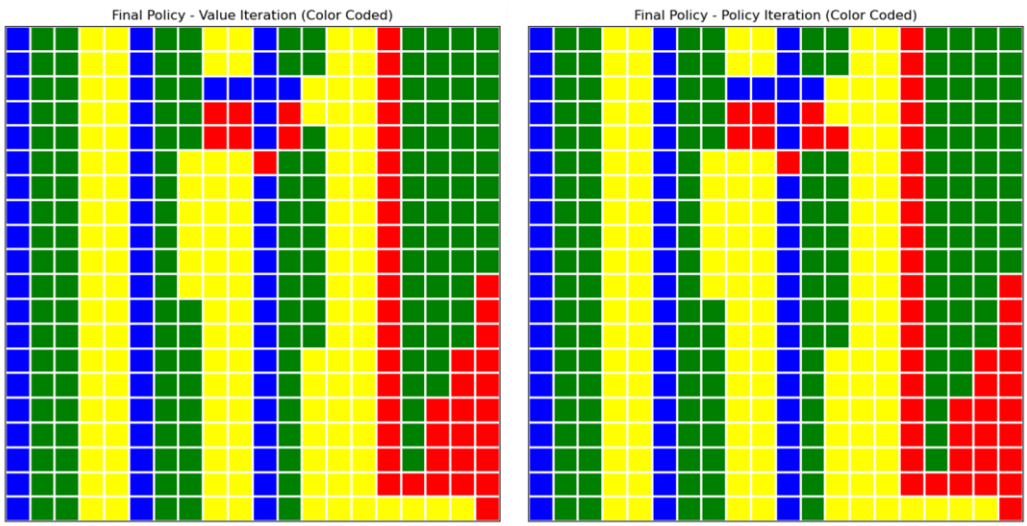

Figure 4 Final Policies obtained from value iteration and policy iteration for Grid world problem. Blue square shows Upward action, Red shows Down, Green shows left, and Yellow shows right

With 400 states in the grid world problem, the computational cost for optimizing policies is significant. In the plot, the spikes in the delta for Policy Iteration suggest that significant changes to the policy occurred at some iterations (around 20, 140, 265). This could be due to the policy facing states where a different action suddenly becomes much more convenient due to updates in the value function of other states. Value Iteration shows a smoother convergence, which is typical because it updates the value estimates in a more continuous fashion. The changes get smaller as the estimated values of the states stabilize. The stochastic states and obstacles within the grid world can cause more sudden shifts in policy. When a policy change happens for a state, it can have a cascade effect on the adjacent states. This can explain the larger spikes observed in the policy iteration delta.

Both policies (Figure 4) show a high degree of similarity, which implies consistency in the solution found by each algorithm. The patterns of actions are almost identical, which is expected if both algorithms have converged to the optimal solution. The red (Down) actions at the top of the grid are notable. This could indicate an environment where moving down from the top initially leads to a more optimal path. It's also possible that these actions represent a strategy to navigate around an obstacle.

Section 2) Implementation of Q-learning

MDP-1 Gambler`s problem

To use Q-learning for Gamblers' problem a random initial state was chosen. Within each episode, the gambler takes actions until the goal is achieved or all money is lost. After each action, the Q-value for the state-action pair is updated based on the reward received and the maximum expected future rewards, considering a discount factor GAMMA and a learning rate ALPHA. The deltas array is used to track the maximum change in Q-values for each episode, which is a common technique to check for convergence of the algorithm. After learning, the final policy is derived by choosing the action with the highest Q-value for each state.

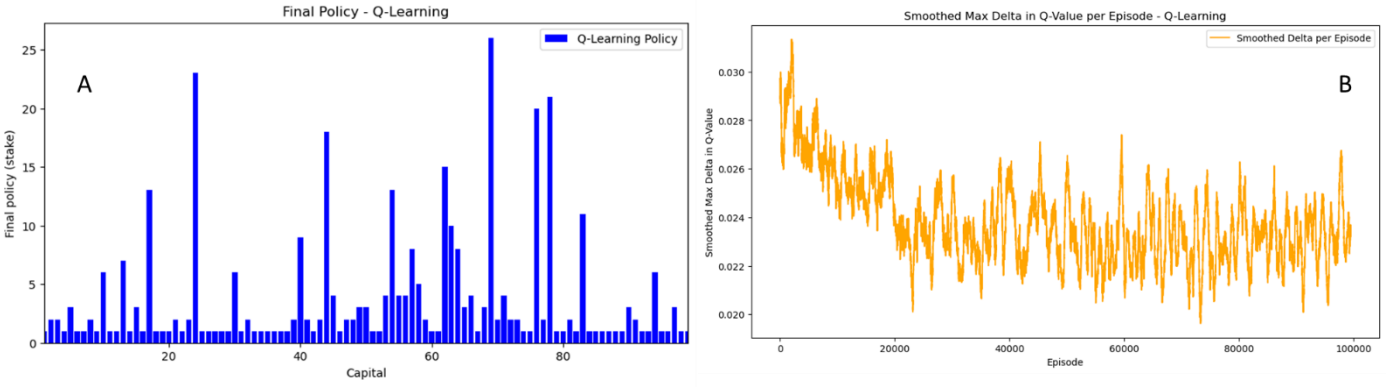

Figure 5-A) Final policy derived from Q-learning. B) Smoothed Maximum delta in Q-value per iteration

Figure 5-A shows the recommended stake for each capital level. The policy suggests varied stakes across different capital levels, with some peaks indicating a preference for larger bets in certain states. This variability in recommended stakes is likely due to the stochastic nature of Q-learning and its exploration strategy. When comparing this to the policies derived from value and policy iteration, we can notice that those methods, which fully know the model, would likely produce smoother policies. However, the Q-learning policy here demonstrates an understanding of advantageous states from which the gambler can make larger bets. Figure 5-B was smoothed using a moving average for a better representation. The plot of smoothed max delta in Q-values indicates how much the Q-values vary from one episode to the next. This graph shows a general downward trend with some fluctuations, suggesting that while the Q-learning algorithm is learning and improving the estimates of the Q-values over iterations, there are still occasional episodes where significant learning occurs. This is possibly due to the exploratory nature of the Epsilon-Greedy strategy. Comparatively, value and policy iteration methods tend to show a smoother convergence as they deterministically improve the policy based on a known model.

The Q-learning algorithm's exploration strategy involves the Epsilon-Greedy method, allowing for both exploration and exploitation during learning. This contrasts with value and policy iteration, which do not require an exploration strategy as they work with a known model and systematically evaluate or improve the policy. The Epsilon-Greedy strategy can introduce more variability in the learned policy, as seen in the peaks and troughs of the policy chart.

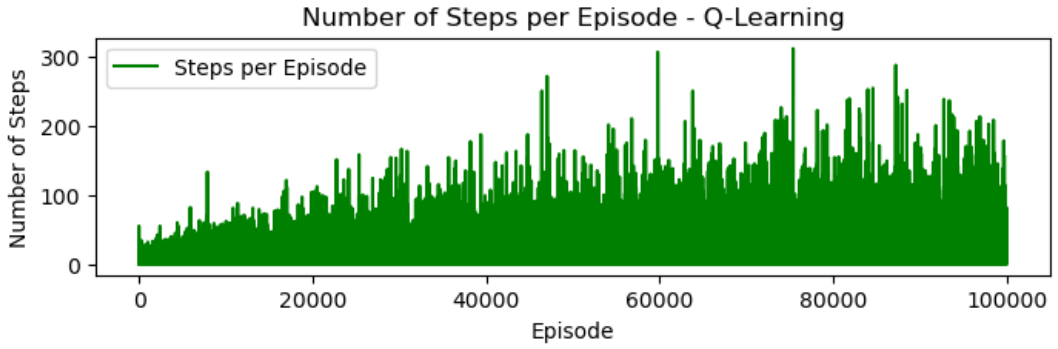

Figure 6. Number of steps per episode in the gambler’s problem

Figure 6 shows the number of steps per episode and provides insight into how the gambler's policy is performing in terms of the length of each gambling session. It shows a wide variation in the number of steps per episode, which may indicate that the learned policy from Q-learning results in a variable path to reaching the goal state. This can be compared to the paths suggested by the policies from value and policy iteration, which are more consistent due to their deterministic nature.

MDP-2 Grid world problem.

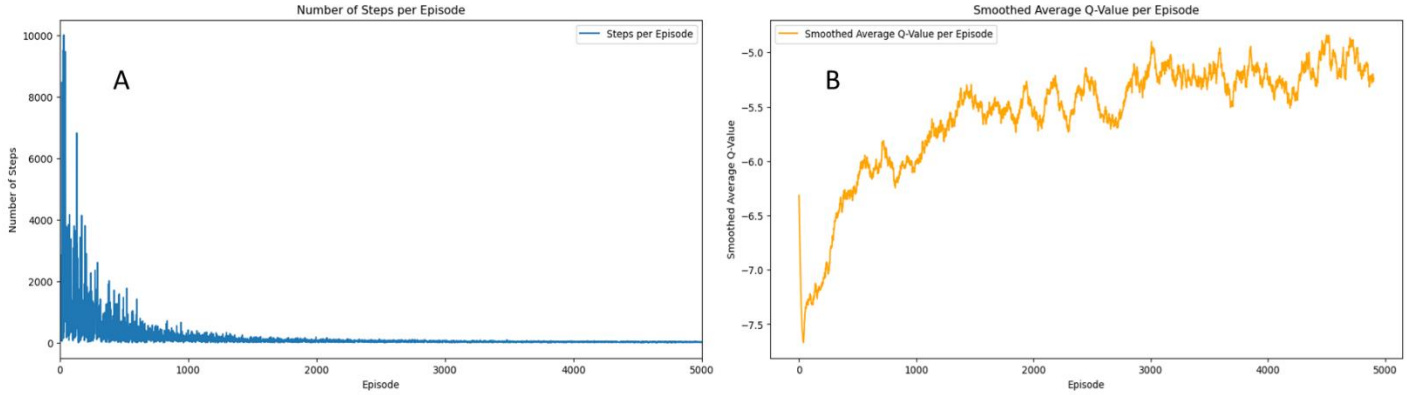

Figure 7-A) Final policy derived from Q-learning and B) Smoothed Maximum delta in Q-value per iteration for grid world problem.

Figure 7-A shows the number of steps per episode giving an indication of how quickly the agent is able to find the goal state as learning progresses. There is a steep decrease initially, which flattens out, suggesting that the Q-learning algorithm quickly learns a reasonably good policy and then refines it over time. Figure 7-B shows the smoothed average Q-value per episode. This graph shows an increase and then a stabilization of the Q-values, suggesting the agent has learned a stable policy.

Figure 8 grid shows the learned policy for each state in the grid world. A consistent pattern of colors would suggest a coherent and possibly optimal strategy to navigate toward the goal state. The variety of colors within the grid indicates that Q-learning has developed a diverse set of actions for different parts of the grid. This is expected, as Q-learning must explore the environment without prior knowledge of state transitions or rewards.

Comparing Q-learning with Value and Policy Iteration:

Convergence: Policy iteration converges in fewer iterations, but each iteration is more computationally expensive due to policy evaluation and improvement steps.

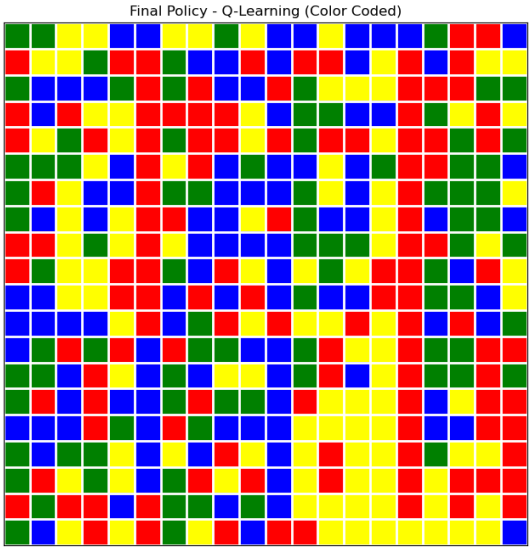

Figure 8 Final policy using Q-learning

Q-learning performs by exploration which is necessary to discover optimal policies but also means that the algorithm spends time performing actions that may not be immediately beneficial to learn about the environment making the algorithm more computationally intensive.

Value iteration converges smoothly as it directly refines the value function, whereas Q-learning shows a quick initial learning phase followed by refinement.

Final Policy: The final policy from value and policy iteration methods is more consistent due to the full knowledge of the model. Q-learning's policy shows more variability but approximates the optimal policy with sufficient exploration and episodes.

Exploration and Learning Dynamics: Q-learning's Epsilon-Greedy, allows it to learn without prior knowledge of the environment, leading to inconsistency compared to value and policy iteration approaches. The peaks in steps per episode and the variations in the average Q-value show that Q-learning's performance can fluctuate as it explores the state space. The solutions derived from value and policy iteration are more consistent, as observed in the action patterns in the grid world problem.

Conclusion

In conclusion, the analysis on the Gambler's Problem and the Grid World Problem revealed distinct specifications and performance metrics for value iteration, policy iteration, and Q-learning algorithms. For the Gambler's Problem, value iteration showed a slower convergence in terms of the number of iterations but was more time-efficient, taking only 0.0358 seconds to converge over 11 iterations. In contrast, policy iteration was faster to converge in terms of iteration count, requiring only 8 iterations, but was less computationally intensive, taking 0.0781 seconds.

The implementation for the Grid World Problem showed similar findings, with value iteration requiring 126 iterations over 33 seconds, and policy iteration taking 26 iterations over 34 seconds. Notably, policy iteration's computational demands for each iteration were higher due to the dual steps of policy evaluation and improvement.

Q-learning provided an explorative element into the learning process, exhibiting a higher variability in the learned policy due to its Epsilon-Greedy exploration strategy. This was evident in the Gambler's Problem, where the final policy suggested different stakes across different capital levels, indicative of the algorithm's dynamic learning and adaptation to the environment.

In the Grid World Problem, Q-learning showed a consistent pattern of actions, suggesting a coherent strategy developed through the exploration of the environment without having prior knowledge of state transitions or rewards. The algorithm’s performance, measured in terms of the number of steps per episode, decreased rapidly, showing an efficient policy formation early in the learning process.

Overall, the study underscored the computational trade-offs between different types of reinforcement learning algorithms. While deterministic methods (value and policy iteration) provide consistency in policies, model-free methods such as Q-learning offer adaptability and resilience in uncertain environments.

References

Sutton, R.S. and Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.

Bellman, R., 1957. A Markovian decision process. Journal of mathematics and mechanics, pp.679-684.