Comparative Analysis of On-Policy and Off-Policy Learning Dynamics in a Simple Grid World Environment

Darcy Corson

November 17, 2023

1 Introduction and Motivation

In the realm of reinforcement learning (RL), Exercise 7.10, presented by Richard S. Sutton and Andrew G. Barto in their text, *Reinforcement Learning: an Introduction*, proposes the construction of a small prediction problem to investigate the relative data efficiencies of two learning methods: a standard off-policy approach, and a simplified off-policy approach. As such, this problem is situated within the framework of off-policy learning, where the main goal is to determine the value of a certain target policy while the agent is engaged in the environment using a different strategy, known as the behavior policy. The key here is the agent's capacity to learn about one specific strategy (the target policy) through the lens of experiences accrued under a different strategy (the behavior policy). This methodology is of particular importance in situations where employing the target policy directly is either not feasible or poses certain risks. All of the experiments described in this paper are set in SimpleGridWorld, which is a streamlined environment that is characterized by a three-state structure.

The initial experiment presented in this paper meets the requirements of the exercise's prompt. That is, it allows for comparison of the performance of a conventional off-policy approach to its more simple counterpart. The distinction in data efficiency between the two approaches was made somewhat evident through observation of the rate of convergence in the value functions derived from each algorithm. This distinction was made more evident by evaluating the relative data efficiencies of the two algorithms and observing their convergence behavior over a range of episode counts. The results of this initial experiment set a foundational understanding of the differing efficiencies of these off-policy algorithms, highlighting the importance of algorithmic choices in RL, especially in scenarios where data efficiency is a key concern.

The subsequent experiments presented in this paper are extensions of the initial experiment and are meant to provide greater insight into the behavior and performance of the off-policy methods being considered. The performance of the SARSA algorithm, an on-policy learning method, was compared to the performance of the off-policy methods, allowing for a direct comparison between the efficiencies of the on-policy and off-policy learning strategies. The off-policy algorithms were also run with various alpha values. The results of this experiment help elucidate how the speed of learning (determined by alpha) influences the efficiency and stability of the off-policy algorithms.

2 Background

2.1 Off-Policy Learning

Off-policy learning involves an agent learning the value of an optimal policy (the target policy) while following a different policy (the behavior policy) during its exploration of the state space. Thus, the target policy in off-policy learning is the policy that the agent ultimately wants to learn about or optimize, and the behavior policy, which may be more exploratory in nature, is used by the agent for interacting with the environment and collecting data. The key technical challenge in off-policy learning is the discrepancy between these two policies. To bridge this gap, off-policy methods often employ importance sampling, a statistical technique that re-weights the experiences gathered under the behavior policy to estimate what they would have been under the target policy. This is achieved by calculating the importance sampling ratio, which is the probability of the trajectories under the target policy divided by their probability under the behavior policy. This ratio adjusts the returns or the value updates to better represent the target policy's perspective.

The iterative update of the value function in off-policy learning can be achieved through various algorithms, each with its unique update rule. For instance, in Q-learning, the update rule is designed to converge towards the optimal policy by always considering the maximum possible reward for the next state, regardless of the policy followed during data collection. This is achieved through the Bellman optimality equation, which iteratively updates the Q-values towards the expected rewards plus the discounted value of the next state under the optimal action.

For off-policy algorithms, because the data is generated under the behavior policy, but the value estimation is for the target policy, the agent is empowered to learn from exploratory actions or historical data generated by a different policy. Thus, the advantage of off-policy learning methods is their ability to leverage an extensive range of experiences, including those from previous iterations, different agents, or hypothetical scenarios. This makes off-policy methods particularly useful in environments where sampling data is expensive or risky, because they can effectively use off-policy data.

In contrast, on-policy learning methods require the agent to learn the value of the policy according to the actions it is currently taking. Thus, for on-policy learning methods, the policy used to make decisions is the same one that is being evaluated and improved. This approach ensures that the policy is always tailored to the agent's latest behavior. An example of an on-policy method is the State-Action-Reward-State-Action (SARSA) algorithm, where the agent learns from the actions it currently takes in the environment. The primary benefit of on-policy learning is that the policy is consistently updated based on the agent's experiences, making it highly adaptive to changing environments. However, this comes at the cost of requiring more data for effective learning, as the agent can only learn from its current strategy.

As mentioned previously, in order to navigate the complexities inherent in off-policy learning, a set of mathematical equations are employed to bridge the gap between the target policy and the behavior policy and accurately estimate the value of the target policy based on the data collected under the behavior policy. These equations form the backbone of off-policy learning algorithms. The choice between using more complex algorithms and simpler versions is made based on the specific requirements and constraints of the learning environment and task at hand.

2.2 Off-Policy Algorithm Using Equations 7.13 and 7.2

An off-policy algorithm that uses Equations 7.13 and 7.2 represents a standard (more complex) approach, utilizing detailed importance sampling for a nuanced alignment of the behavior policy's returns with the target policy.

$$G_{t:h} = \rho_t(R_{t+1} + G_{t+1:h}) + (1 - \rho_t)V_{h-1}(S_t), \quad t < h < T, \tag{7.13}$$

Equation (7.13) is used to calculate the off-policy return, which is the total expected reward, with the aid of importance sampling. Specifically, the equation calculates the return $G_{t:h}$ for a state at time t while accounting for future rewards and the value of future states. It incorporates the importance sampling ratio ρ_t, which adjusts the returns based on the probability of taking actions under the target policy as opposed to the behavior policy. The equation aims to estimate returns as if the target policy were followed, counterbalancing the difference between the two policies.

$$V_{t+n}(S_t) = V_{t+n-1}(S_t) + \alpha\left[G_{t:t+n} - V_{t+n-1}(S_t)\right], \quad 0 < t < T, \tag{7.2}$$

Equation (7.2) describes how to update the value of a state in a way that incorporates new information obtained from recent experiences. For this equation, the value of a state S_t at time t is updated using the return calculated from Equation (7.13). The learning rate α dictates the degree to which new information influences the value update. This gradual adjustment of value estimates aligns them more closely to the returns computed under the target policy.

2.3 Off-Policy Algorithm Using Equations 7.1 and 7.9

An off-policy algorithm that uses Equations 7.1 and 7.9 represents a simplified methodology. While this approach may sacrifice some degree of accuracy, it benefits from computational efficiency while still incorporating importance sampling for policy correction.

$$G_{t:t+n} = R_{t+1} + R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n}), \tag{7.1}$$

Equation (7.1) is a simplified method for calculating the return in off-policy learning scenarios. Specifically, this formula computes the return $G_{t:t+h}$ from time t to $t{+}n$, encompassing the sum of rewards up to t+n and the discounted value of the state at t+n. Because this more simple return calculation may not include the complexities of importance sampling, it is more computationally efficient.

$$V_{t+n}(S_t) = V_{t+n-1}(S_t) + \alpha \rho_{t:t+n-1} \left[G_{t:t+n} - V_{t+n-1}(S_t) \right], \quad 0 \le t < T, \tag{7.9}$$

Equation (7.9) is a more simple approach to updating the value of a state, while still incorporating an element of importance sampling. In this equation, the state value S_t is updated using the return $G_{t:t+h}$ but with a simplified importance sampling correction factor $\rho_{t:t+n-1}$. This adjustment modifies the influence of the return based on the policy discrepancy, though in a less complex way as compared to Equation (7.13).

2.4 The Role of Learning Rate in Off-Policy Algorithms

For off-policy algorithms, the learning rate, α, can greatly influence how quickly and effectively an algorithm learns, how stable its learning trajectory is, and how well it adapts to new information and changing environments. The optimal setting of α can be difficult to discern, and is shaped by the specific characteristics of the learning algorithm, the nature of the environment, and the desired balance between rapid learning and stability. As such, careful tuning of α, and potentially its dynamic adjustment, is critical to the successful implementation of off-policy reinforcement learning algorithms.

A higher α signifies aggressive learning, where recent experiences substantially influence the value estimates. This approach can be particularly advantageous in dynamic environments, where the agent needs to rapidly adapt to changing conditions or newly discovered information. However, a high learning rate comes at the potential cost of instability, as it can lead to significant oscillations in the value function. This is especially true if the new experiences are outliers or are noisy. A lower α promotes stability by giving more weight to the accumulated knowledge, thus dampening the impact of new information. While this ensures smoother convergence and less susceptibility to data variability, it can hinder the learning process, especially in complex environments where the agent might require faster adaptation to optimize its policy.

For the two off-policy algorithms being considered in this paper, α plays a crucial role in the algorithm's convergence to an optimal policy. This significance is encapsulated in the value update equation used in these algorithms:

$$V(S_t) \leftarrow V(S_t) + \alpha \times \text{AdjustmentTerm}.$$

This equation shows the role of in updating the estimated value $V(S_t)$ of a particular situation or state in the learning process. Essentially, α determines how much this estimated value should change based on new information the algorithm receives. This process determines how the algorithm gradually improves its decision-making strategy.

In an ideal theoretical scenario, α should gradually decrease as the algorithm learns more. This means that in the beginning, the algorithm makes larger adjustments to its strategy, but as it becomes more knowledgeable, it makes smaller tweaks. This gradual reduction in α helps the algorithm fine-tune its strategy until it finds the best possible one (known as the optimal policy). However, in real-world applications, keeping α fixed at a smaller value is often preferred. This approach is a balance between two goals: quickly adapting to new information (rapid adaptation) and not changing too drastically with each new piece of information (stability). If α is too high, the algorithm might change its strategy too radically and too often, which can lead to instability and/or erratic behavior. If α is too low, the algorithm might learn too slowly or get stuck with a sub-optimal strategy. Using a fixed, smaller α makes the algorithm easier to manage and implement, but it also means that α needs to be set very thoughtfully. If it's not set correctly, the algorithm might not learn the best strategy (converge to a sub-optimal policy) or it might not be able to finalize its learning process (non-convergence).

2.5 Policy Stability Over Time for Off-Policy Algorithms

As discussed, off-policy algorithms are unique in their ability to learn an optimal policy that is different from the one under which the agent currently operates. This characteristic highlights the importance of policy stability, which refers to the consistency and reliability of the policy learned by an agent over time. A stable policy implies that the value function estimates and subsequent actions of the agent do not undergo drastic changes after a certain point in the learning process, indicating convergence to an optimal strategy. Achieving policy stability in off-policy learning is challenging due to the inherent separation between the behavior policy and the target policy. This separation can lead to discrepancies in the learning process, making it harder for the algorithm to stabilize at an optimal policy. This problem may be compounded in environments with high variability or sparse rewards.

3 Experiments

3.1 SimpleGridWorld Environment

For each experiment performed, learning algorithms were run on a simple grid world environment, SimpleGridWorld. SimpleGridWorld is a basic implementation of a grid-like environment consisting of a set of states, [0, 1, 2]. In a more complex grid world, these might correspond to different positions on a grid, but in this simplified version, they are just abstract states with linear transitions. State 2 is designated as the ”end state”, meaning that it is the terminal state. Reaching this state concludes an episode. The ”step” method defines the logic for transitioning from one state to another given the current state and an action. In this environment, actions are not explicitly defined because the transition is deterministic. That is, taking an action in state 0 or 1 always moves the agent to the next state (e.g. from 0 to 1 or from 1 to 2). When the agent is in the terminal state (2), any action will keep the agent in the same state, effectively ending the episode. Each transition incurs a reward of -1, except transitions within the terminal state, which incur a reward of 0. This setup encourages the agent to reach the terminal state efficiently because it minimizes the negative reward. The ”simple behavior policy” function randomly chooses between two actions (labeled 0 and 1) for each state. ”Simple target policy” is the policy that the agent is actually trying to learn and improve. Here, it’s a static policy that always returns action 1, regardless of the state.

The SimpleGridWorld environment used here, while useful for controlled experiments, is significantly less complex than real-world environments. As such, it may not capture the complexity, unpredictability, and diversity of states and actions encountered in real-world applications. Findings derived from such a simplified model may not directly translate to more complex environments.

3.2 Original Experiment

The original experiment compares the relative data efficiency of the two off-policy reinforcement learning algorithms discussed in the ”Background” section of this paper: the standard (more complex) off-policy algorithm using Equations (7.13) and (7.2), and the more simple off-policy algorithm using Equations (7.1) and (7.9). The experiment is conducted the SimpleGridWorld environment. As discussed above, the SimpleGridWorld environment consists of three states, [0, 1, 2], where state 2 is a terminal state.

3.2.1 Procedure

In each episode of the experiment, the agent starts at state 0 and makes transitions until it reaches the terminal state. The step function dictates the environment dynamics, where each action taken in a non-terminal state transitions the agent to the next state with a reward of -1, and any action taken in the terminal state keeps the agent in the same state with a reward of 0.

The standard off-policy algorithm uses importance sampling to account for the difference between the behavior and target policies. Importance sampling weights (W) are applied to correct for the discrepancy in the probabilities of the behavior policy and the target policy. The update rule for the value of a state is:

$$V(s) = V(s) + \alpha \cdot W \cdot (G - V(s)).$$

Here, G is the cumulative reward from the current state to the terminal state, W is the importance sampling weight, and α is the learning rate.

The simplified off-policy algorithm removes the importance sampling component and updates the value only when the behavior policy's action matches the target policy's action (denoted by $\rho = 1$ or 0). The update rule here is:

$$V(s) = V(s) + \alpha \cdot \rho \cdot (G - V(s)).$$

The experiment runs multiple trials of these algorithms to average out the variability due to the stochastic nature of the behavior policy. The "run multiple trials" function takes an algorithm, a target policy, and a behavior policy, and runs the specified algorithm for a number of episodes, repeating this over many trials to get an average value function over time. The experiment then plots the average convergence of the value function over episodes for both algorithms. This comparison aims to demonstrate the effect of using importance sampling in off-policy learning and to show how the value function estimates converge over time when averaged over multiple trials.

The experiment then ventures beyond this preliminary assessment and conducts an exploration of the convergence patterns of the two algorithms given different amounts of data. Data efficiency is a measure of an algorithm's capability to harness information from available data to enhance its performance. In this context, the available data is quantified as episodes of experience. That is, each episode represents an iteration of experience from which the algorithm can extract knowledge and refine its decision-making process. Thus, this part of the experiment evaluates not just the final outcome but also the progression towards convergence, observing the rate at which performance improvements diminish and stabilize within a predefined threshold of learning stability.

The "calculate convergence time" function is a diagnostic tool to pinpoint the episode at which the algorithm's value function stabilizes within a user-defined threshold, indicating convergence. The "test convergence" function orchestrates the experiment by running the algorithms through multiple trials for a range of episode counts and utilizes "calculate convergence time" to determine the convergence time for each count. It compiles these into a dictionary, mapping each episode count to its corresponding convergence time.

The experiment proceeds to define a set of episode counts to test – 10, 100, 250, 500, 750, and 1000 – covering a broad spectrum from early to advanced learning stages. A very fine convergence threshold is set, representing a stringent criterion for learning stability. The resulting convergence times are plotted on a graph with the episode count on the x-axis and the episodes to convergence on the y-axis, providing a visual representation of the algorithms' efficiency across the episode spectrum.

3.2.2 Results

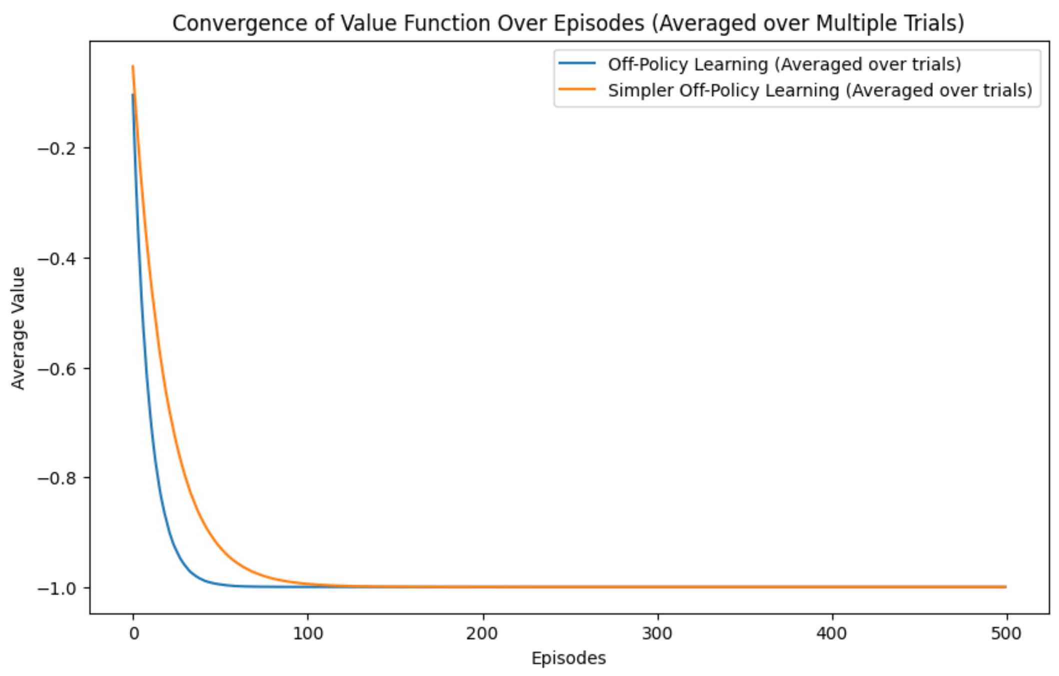

Figure 1: The graph displays the average convergence of two distinct off-policy reinforcement learning algorithms over 500 episodes, with the results averaged over 1000 trials. The blue curve represents an off-policy learning algorithm that utilizes importance sampling, while the orange curve corresponds to a simpler version of the off-policy learning algorithm that does not use importance sampling. The convergence of the importance sampling algorithm is slightly faster, suggesting a more efficient learning process.

The graph in Figure 1 illustrates the results of the first part of the original experiment. The x-axis represents the number of episodes, and the y-axis represents the average value, which is the average of the value function estimates across all states and trials.

Both curves show a sharp decline initially, which indicates that the value estimates for the states are quickly decreasing. This rapid change is common early in learning when the initial value estimates (which are frequently optimistically initialized) are adjusted significantly based on the actual rewards received from the environment. The decline slows down as the number of episodes increases, demonstrating the convergence of the value function as the agent learns from the environment.

The blue curve represents the standard off-policy learning algorithm with importance sampling, and the orange curve represents the more simple off-policy learning algorithm without importance sampling. Both algorithms start with similar value estimates, but as learning progresses, we observe that the two algorithms converge at different rates. Initially, both algorithms appear to learn quickly, but the convergence of the more simple algorithm, without importance sampling, is slightly slower. This suggests that while both algorithms improve their value estimates over time, the inclusion of importance sampling may provide a more accurate or efficient learning process. Both curves approach stability towards the later episodes, flattening out as they converge to a steady-state value function. This behavior suggests that the agent is reaching a policy that consistently estimates the expected returns from each state under the target policy.

It's important to note a few additional things about this graph. First, the average value remains negative, which aligns with the environment's reward structure, where the agent receives a negative reward for each step until it reaches the terminal state. The goal in such an environment is to find a policy that minimizes the negative reward, which, in this case, would be reaching the terminal state as quickly as possible. Second, the smoothness of the curves is a result of averaging the outcomes over a large number of trials, mitigating the variance inherent in the stochastic policy and transitions. This smoothness indicates that the results are statistically robust, reflecting the general performance of the algorithms rather than the outcome of any individual or outlier trial.

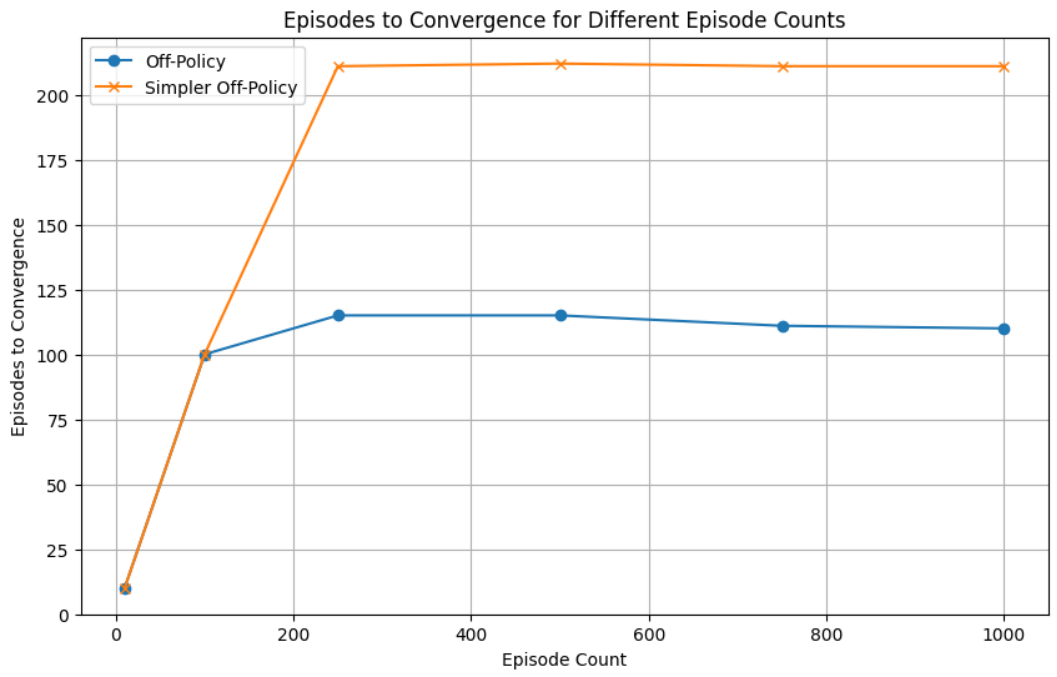

Figure 2: The graph illustrates the number of episodes required to reach convergence for two different reinforcement learning algorithms across varying episode counts. The off-policy algorithm's convergence times show a decreasing trend as the episode count increases, suggesting improved efficiency or stability with more episodes. In contrast, the simpler off-policy algorithm requires a significantly higher number of episodes to converge, especially noticeable in the initial stages, but also levels off as the episode count grows. This suggests that the off-policy algorithm is more data-efficient.

The graph in Figure 2 illustrates the results of the second part of the original experiment. That is, it provides a comparative analysis of the convergence times for the two reinforcement learning algorithms as they process and learn from a set range of episodes. The x-axis enumerates the total count of episodes, serving as a measure of experience or data available for the algorithms to learn from. The y-axis quantifies the number of episodes each algorithm takes to reach a point of convergence, which is defined as the moment when the algorithm's value function ceases to show significant changes (less than 0.000001).

For both algorithms, an initial increase in the number of episodes required for convergence is observed. This could be due to the algorithms not having reached a stable learning phase with the limited episodes. This could be a feature of the algorithms themselves or a characteristic of the environment in which they are operating, suggesting that there is a minimum threshold of experience necessary before meaningful learning can occur. Once past this threshold, the number of episodes to convergence should decrease, reflecting a more efficient learning process as the algorithm begins to utilize the data effectively.

The standard off-policy algorithm, in blue, displays a learning curve that begins with a rapid increase in the number of episodes required for convergence as the episode count increases from 10 to 100. This trend flattens out as the episode count continues to rise, indicating a plateau in learning efficiency. This plateau suggests that the algorithm's learning performance stabilizes and that providing additional episodes beyond a certain point does not significantly speed up convergence. This behavior can be interpreted as the algorithm achieving an optimal learning state relatively quickly, after which additional data yields diminishing returns in terms of learning speed.

The simpler off-policy algorithm, in orange, also shows a steep ascent in the initial phase, where the episodes to convergence dramatically increase to an even higher episode count. This ascent suggests that the algorithm struggles to learn efficiently from the given data, requiring more episodes to converge as the episode count grows. Similar to the off-policy algorithm, the simpler off-policy's curve also begins to level off, indicating that it eventually reaches a point where additional episodes do not correspond to faster convergence.

The most important feature of the graph is the wide divergence in the performance of the two algorithms. The off-policy algorithm requires about half as many episodes as the simpler off-policy algorithm does to converge, and improves over time, suggesting a more robust learning strategy or more efficient use of the experience data. The simpler off-policy algorithm's performance lag indicates inefficiencies in learning or a need for more data to achieve similar levels of understanding and optimization.

3.3 Extension Experiment 1: Comparing Performance of Off-Policy Algorithms to Performance of SARSA

The first extension experiment involves implementing and comparing the SARSA on-policy learning algorithm against the two off-policy learning methods discussed above using the SimpleGridWorld environment.

3.3.1 Procedure

The SARSA algorithm, an on-policy method, is used to estimate the quality (Q-value) of each action in each state. The Q-values represent the expected utility of taking an action in a given state and following the policy thereafter. In this experiment, the algorithm initializes a Q-value table with zeros, iterates through a set number of episodes, and updates the Q-values using the SARSA update rule given by:

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

Here, s and a represent the current state and action, s' and a' are the next state and action, r is the reward received after taking action a in state s, α is the learning rate and γ is the discount factor. The policy used to select actions during learning is epsilon-greedy, which chooses a random action with probability ϵ to encourage exploration, or the action with the highest Q-value otherwise to exploit the agent's current knowledge. "The sarsa behavior policy" function implements this epsilon-greedy policy, deciding the action based on the current Q-values, and an exploration factor ϵ.

This experiment records the average of the Q-values across all states and actions after each episode to observe the convergence of the learning process over time. This is compared with the results from the previously implemented off-policy methods, which also aimed to converge to the optimal value function via different strategies. The results are plotted to visually compare the performance and convergence rate of the SARSA algorithm against the off-policy methods. The subsequent graph provides insight into the dynamics of on-policy versus off-policy learning in a controlled setting.

3.3.2 Results

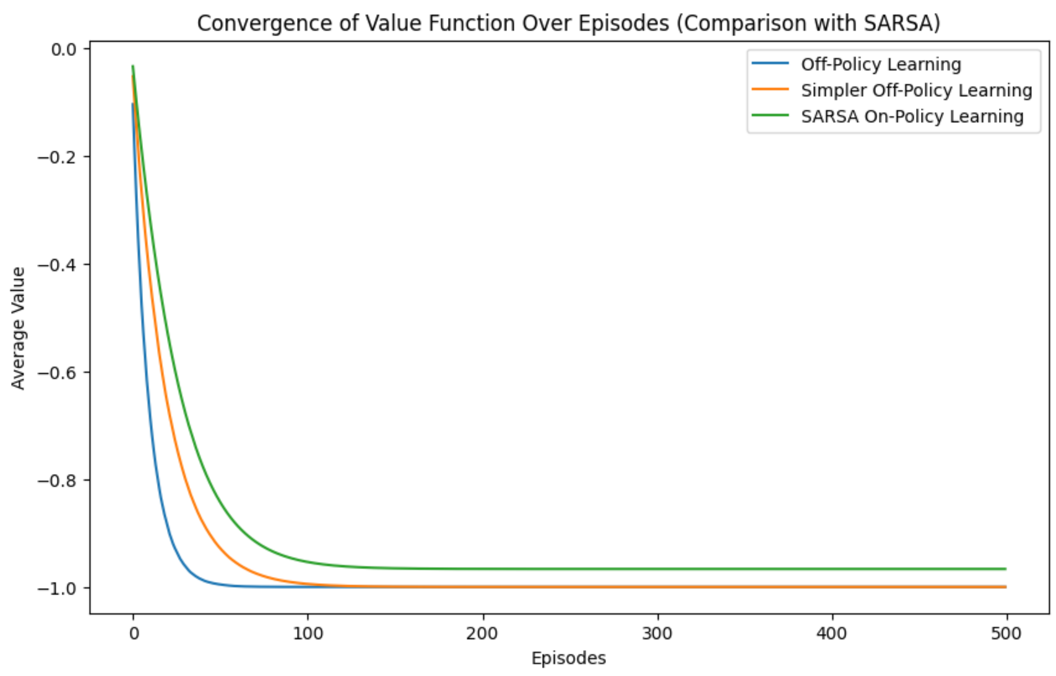

Figure 3: The graph illustrates the convergence trajectories of three reinforcement learning algorithms in a simple grid world environment, measured over 500 episodes and averaged across 1000 trials. The standard off-policy learning (blue) and simpler off-policy learning (orange) algorithms are compared to the SARSA on-policy learning algorithm (green), and each curve depicts the rapid initial learning phase followed by stabilization as the algorithms converge to their respective optimal policies. The convergence paths indicate that off-policy learning methods exhibit a faster initial rate of learning, suggesting a more efficient learning process.

The graph in Figure 3 illustrates the results of the first extension experiment. The y-axis represents the average value, which indicates the expected return from the states under the policies learned by each algorithm. The x-axis represents the number of episodes, each episode being a sequence of actions and state transitions that the agent undergoes until it reaches a terminal state.

All three algorithms show a steep initial learning curve, quickly improving the value function estimates from their initial values. This steep curve suggests that the agent is rapidly learning from its initial exploration, significantly updating its policy based on the new information it gathers from the environment's rewards. The SARSA on-policy learning curve (green) closely follows the off-policy learning curves (blue and orange) but seems to converge more slower. SARSA is an on-policy method, meaning that it evaluates and improves the policy that it derives its behavior from. In contrast, the off-policy methods evaluate a policy different from the one that guides their behavior. The off-policy methods may initially learn faster because they have the potential to utilize a more optimal policy for their updates. As the number of episodes increases, the average value for all methods converges towards a stable value. This plateau suggests that the algorithms are reaching an optimal policy where the agent has learned to navigate the environment effectively, accruing the maximum possible return from any given state.

The similarity in the convergence paths of the three algorithms indicates that they are all effective in this simple environment. The slight differences in the rate of convergence and the final converged values could be due to the intrinsic properties of the SARSA algorithm compared to the other off-policy methods. That is, SARSA takes into account the current policy's action (including the exploration steps) for its updates, which could lead to a more conservative learning path.

It is interesting to note that the graph seems to show that the value functions for the three reinforcement learning algorithms, standard off-policy, simpler off-policy, and SARSA on-policy, are not converging to the same value. Several reasons could explain these differing convergence points. First, the intrinsic differences between the algorithms' policies could be a factor. Off-policy methods are designed to evaluate a target policy that is different from the behavior policy they use to explore the environment. In contrast, SARSA is an on-policy method that evaluates and improves the policy according to which it makes decisions. This fundamental distinction means that the off-policy methods and SARSA are essentially optimizing different objectives, which may lead to different convergence values. Further, the algorithms' different parameter settings may be contributing to this difference. Different learning rates α and different discount factors γ could lead to different rates of convergence and final converged values. Initial value estimates can also influence the algorithms' learning trajectories; starting from different initial values could lead to different converged values.

3.4 Extension Experiment 2: The Effect of Varying the Learning Rate on the Convergence of Off-Policy Learning Algorithms

The second extension experiment involves exploring the impact of varying the learning rate α on the convergence of the two off-policy learning algorithms within SimpleGridWorld.

3.4.1 Procedure

In each episode, the agent starts from the initial state and makes transitions within the environment until the terminal state is reached. The cumulative reward G is updated at each step, and the value function V for each state is updated using the respective rules for each algorithm, as discussed above. The experiment is conducted with different values of alpha: [0.01, 0.1, 0.5], representing low, medium, and high learning rates. These rates control how quickly the algorithm adjusts the value estimates in response to new information. The "run experiment with varying alpha" function runs the experiments for both algorithms across the specified alpha values. The average values of the value function over time for each α are calculated and stored.

The results of this experimentation are plotted in two separate graphs – one for the standard off-policy algorithm and another for the simpler off-policy algorithm. Each graph shows the convergence patterns of the algorithms over 500 episodes, with different lines representing different α values. These plots illustrate how the value functions' convergence rates differ with the change in the learning rate. A lower α is expected to show a smoother but potentially slower convergence, while a higher alpha might show faster but more volatile convergence.

3.4.2 Results

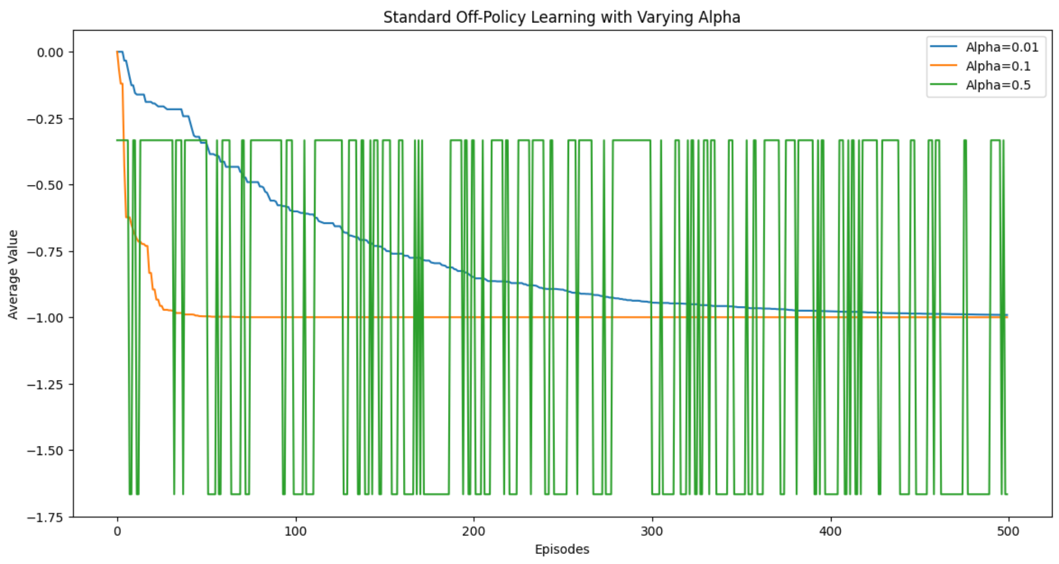

Figure 4: This figure illustrates the average value function convergence for the off-policy algorithm over 500 episodes for three different learning rates. The blue line (alpha = 0.01) shows a gradual and stable convergence, indicating a conservative update approach. The orange line (alpha = 0.1) finds a middle ground, with steadier convergence and less volatility, suggesting an optimal learning rate. The green line (alpha = 0.5) exhibits significant fluctuations, reflecting instability due to overly aggressive value updates.

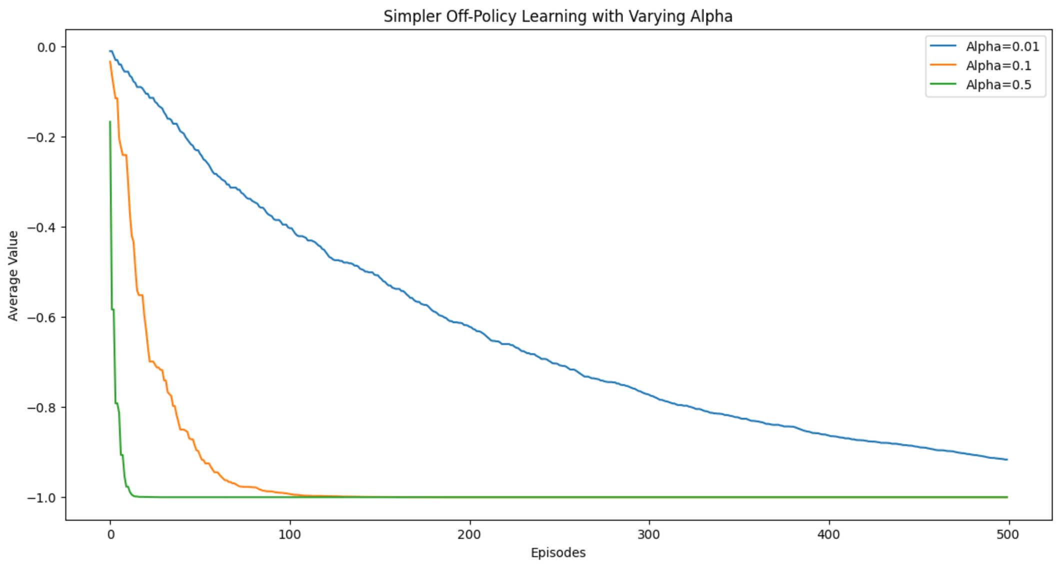

Figure 5: This figure depicts the evolution of average value function estimates over 500 episodes for a simpler off-policy reinforcement learning algorithm. Each line represents the trajectory for a distinct learning rate, with alpha = 0.01 (blue) showing a consistent yet gradual convergence, alpha = 0.1 (orange) demonstrating a faster convergence that stabilizes after initial fluctuations, and alpha = 0.5 (green) quickly reaching and maintaining a plateau, suggesting a potentially sub-optimal policy due to overly rapid updates.

The graphs in Figure 4 and Figure 5 illustrate the results of the second extension experiment by displaying the progression of average value estimates across episodes for off-policy learning algorithms, with three different learning rates applied: 0.01 (blue), 0.1 (orange), and 0.5 (yellow). These values represent how aggressively the algorithm updates its value estimates in response to new information. The x-axis indicates the number of episodes, while the y-axis shows the average value across all states.

Figure 4 illustrates these results for the standard off-policy learning algorithm. From this graph, we observe that the learning rate significantly influences the stability and convergence of the value function estimates. The curve corresponding to the highest learning rate ($\alpha = 0.5$) shows a great deal of volatility, with the average value fluctuating widely. This suggests that a high learning rate may be too aggressive, causing the value estimates to overshoot and fail to settle into a stable pattern. The plots for α values of 0.01 and 0.1 exhibit smoother trajectories, indicating more stable learning processes. However, the curve for $\alpha = 0.01$ converges more slowly than for $\alpha = 0.1$, which is expected because a smaller α leads to smaller updates to the value estimates and thus requires more episodes to converge.

In Figure 4, there is a sharp decline at the beginning of the learning process for all three α values, which is typical as the initial value estimates are likely to be optimistic and are quickly corrected as the agent begins to interact with the environment. After the initial episodes, the curves for $\alpha = 0.01$ and 0.1 begin to flatten, suggesting that the value estimates are approaching a stable policy. In contrast, the $\alpha = 0.5$ curve does not demonstrate a clear stabilization, which may indicate that the policy is not converging effectively within the given number of episodes due to the high learning rate.

Figure 5 illustrates the results of this experiment for the more simple off-policy learning algorithm. Again, the graph plots the average value against the number of episodes for three different α values: 0.01, 0.1, and 0.5, depicted by blue, orange, and green lines, respectively.

For Figure 5, The blue line, corresponding to the lowest α of 0.01, indicates a conservative update approach. It shows a slow and steady improvement in the value estimate, which suggests that the agent incrementally learns from its experiences. This approach, while stable, may require a substantial number of episodes to converge to an optimal policy, as the value function's slow ascent reflects. The orange line, representing an α of 0.1, demonstrates a quicker convergence to a stable policy. The steeper initial slope indicates more significant updates to the value function, allowing the agent to learn more rapidly from its interactions with the environment. The line's subsequent plateau suggests that it stabilizes after the initial learning burst, likely converging to a near-optimal policy. The green line, with the highest α of 0.5, shows an even steeper initial descent but quickly flattens out, maintaining a consistent average value throughout most of the episodes.

In the context of the simpler off-policy learning algorithm, the convergence of the three lines to a plateau within the plot illustrates that, regardless of the initial learning rate, the algorithm has reached a point where further learning does not significantly change the value estimates. That is, the algorithm has learned as much as it can about the environment given the current policy and is now consistently estimating the value function across all states.

From the analysis of both Figure 4 and Figure 5, several conclusions can be drawn about the the impact of α on simple versus more complex off-policy algorithms. For simpler off-policy algorithms, which typically involve fewer calculations and less complexity in updating the value estimates, changing α has a more direct and observable impact on learning performance. A low α leads to slow convergence, which can be useful in noisy environments but may also result in the algorithm failing to converge within a reasonable number of episodes. An intermediate α seems to offer an optimal balance, allowing for steady learning and convergence to a stable policy without the overshoots and oscillations associated with higher α values. A high α can cause the value function to converge quickly, but there is a risk of instability or convergence to a sub-optimal policy due to over-fitting to recent rewards or noise. For more complex off-policy algorithms, the choice of α may significantly impact learning performance, but the relationship might not be as straightforward. For these algorithms, a low α may lead to very slow learning rates and possibly poor performance if the algorithm cannot adequately capture the environment's dynamics within the allotted episodes. An intermediate α might still provide a good balance, but the optimal value for α might shift due to the interactions between the learning rate and the other components of the algorithm. A high α might be especially problematic in complex algorithms as it might amplify any instability.

4 Other Extensions

Expanding the original experiment discussed in this paper to include scenarios of partial observability would be an interesting extension of this work in off-policy RL. In such an extended investigation, the SimpleGridWorld could be modified into a Partially Observable Markov Decision Process (POMDP), which is a situation where the agent has limited access to the state information of the environment. This shift models real-world situations where decision-making frequently occurs when things are uncertain, and full knowledge of the environment isn't always available. Thus, this experimental extension would provide valuable insights into how off-policy algorithms perform under conditions of uncertainty and limited information. Further, it could involve examining whether these algorithms can still effectively learn the value function or policy when they have to deal with ambiguity about the state of the environment.

5 References

Sutton, Richard S., and Andrew G. Barto. Reinforcement Learning: An Introduction. The MIT Press, 2018.