Manny Cassar
March 30, 2025
CISC 474
20213773

Reinforcement Learning Assignment 2

1. Introduction

The goal of this assignment is to implement and compare four different reinforcement learning algorithms within a Gridworld environment; on and off-policy approaches for Monte Carlo and Temporal Difference learning. To accomplish this, I tracked and plotted each run of each algorithm, providing insight into how each approach learns optimal policies and state-action values (Q-values). To start, I constructed a 3×4 grid where my agent starts at the bottom-left cell and must navigate toward the goal cell while avoiding danger cells and blocked cells. Every non-terminal move incurs a small negative penalty -0.1, reaching the goal awards +1, and stepping into a danger cell results in -1.

2. Methodology

2.1 Environment and Setup

I implemented this environment using the GridWorld class from assignment 1. The enumeration for actions, defined in Action(IntEnum), allows the agent to choose moves up, right, down, and left in each state.

The constructor takes parameters such as height, width, start, goal, and lists for danger and blocked cells. I adjusted these parameters to encode the grid described by the assignment description. The methods reset(), get_reward(), and _state_from_action() are used to initialize the grid, return the appropriate reward for a given state, and handle state transitions (including handling of walls and blocked cells), respectively.

2.2 On-Policy Monte Carlo Control

In my on_policy_montecarlo_control() function, I generated full episodes using an ϵ-greedy policy. Each episode is recorded as a sequence of tuples (state, action, reward), and I used first-visit MC updates to compute the return for each state-action pair. I stored these returns in a nested list $(R[state][action])$ and updated the Q-values by averaging these returns. After each episode, I improved the policy by setting the best action to have a probability of $1 - \epsilon + \frac{\epsilon}{N}$ and all other actions to $\frac{\epsilon}{N}$ (where N is the number of actions). This implementation is found in the code block for on_policy_montecarlo_control().

2.3 Off-Policy Monte Carlo Control

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). The MIT Press.

My off_policy_montecarlo_control() function uses ordinary importance sampling to correct for the differences between a fixed behavior policy and the target policy. I generated episodes using a behavior policy that favored upward and rightward moves to help guide the agent toward the goal. As I processed each episode in reverse order, I updated the cumulative importance sampling weights (stored in $C[state][action]$) and adjusted the Q-values accordingly. This method allowed me to update the target policy only when the taken actions were consistent with the greedy choice, as seen in the code.

2.4 On-Policy TD Control (SARSA)

In my on_policy_td_control() method, I applied the SARSA algorithm. I updated the Q-values incrementally after every step using the update rule:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right)$$

I ensured that the policy was updated to be ϵ-greedy after each episode, using the Q-values computed during the episode. The method's code shows how I sample actions based on the current policy, record the (state, action, reward) tuple, and update Q-values on-the-fly.

2.5 Off-Policy TD Control (Q-Learning)

My off_policy_td_control() function implements Q-learning, where I update the Q-values using the maximum Q-value of the next state rather than the Q-value corresponding to the action taken. The update rule used is:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right)$$

This off-policy update allows the algorithm to learn the optimal policy more aggressively. Although the exploration is still handled by an ϵ-greedy behavior, the target for the update is always the greedy action's value.

2.6 Parameter Settings

For my experiments, I ran each algorithm with multiple hyperparameter combinations:

- 1000 and 10,000 episodes
- Discount factor γ to 0.95 and 0.99
- Exploration rate ϵ to 0.05 and 0.1
- Step-size parameter α to 0.1 and 0.2 (where applicable).

These parameters were passed to the respective functions to maintain consistency across the different approaches.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). The MIT Press.

Manny Cassar
March 30, 2025
CISC 474
20213773

---

3. Experiment Results

3.1 On-Policy Monte Carlo (Every-Visit)

Optimal Policy Achievements

Two hyperparameter configurations yielded optimal policies:

- **Configuration A:** Discount Factor (df) = 0.95, ε = 0.05, and 10,000 episodes
- **Configuration B:** Discount Factor (df) = 0.99, ε = 0.10, and 10,000 episodes

With the first configuration, a lower ε promotes a more deterministic behavior as the agent's actions are less influenced by exploratory randomness. This setting, in tandem with a moderate discount factor, biases the agent toward immediate rewards. In contrast, the second configuration employs a higher discount factor , which values long-term rewards more heavily, coupled with a slightly increased exploration rate. This combination allows the agent to gradually learn a more global view of the environment. The policy converges towards the optimal behavior through a more exploratory and initially diffuse process. All other combinations of hyperparameters resulted in sub-optimal policies.

Smoothing of State Values with Increased Episodes

When I analyzed the heatmaps for 1000 episodes, I noticed that some non-goal states exhibited a value of close to 0, indicating that the Q-value estimates had not yet propagated meaningful reward information. Monte Carlo methods rely on full-episode returns, and with limited episodes, there is insufficient data to cover all state-action pairs adequately.

However, as the number of episodes increased to 10,000, the heatmaps transformed into a much smoother gradient. This smoothing occurs because the variance decreases and the estimated state values have more Q updates to converge to their true expected returns.

Trade-Offs and Comments

Discount Factor: A higher discount factor promotes long-term planning and smoother value gradients, but may require more iterations to stabilize. I found that the size of this grid means that a slightly lower discount factor converges more quickly, especially when combined with lower exploration.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). The MIT Press.

Exploration Rate: Lower ε accelerates convergence through exploitation, potentially limiting early exploration. Higher ε encourages broader exploration, initially causing noisier policies but ultimately supporting robust global policy convergence.

Number of Episodes: Higher episode counts improve data coverage, reduce variance, and create clearer value gradients, enhancing policy accuracy. The heatmaps from 1000 to 10000 episodes clearly demonstrate that sample efficiency is a key challenge in Monte Carlo methods. Fewer episodes lead to sparse data and zero-valued state estimates in some areas of the grid, whereas a larger number of episodes allows for a comprehensive evaluation of the state space, resulting in a smoother and more accurate heatmap.

3.2 Off-Policy Monte Carlo (Every-Visit)

Hyperparameter Combinations and Convergence to the Optimal Policy

In my experiments with off-policy Monte Carlo control, I observed that all hyperparameter combinations tested eventually resulted in the discovery of the optimal policy, underscoring off-policy robustness despite variance from importance sampling corrections.

Gradient Differences and Training Iterations

I noticed that as the number of training episodes increased, the gradient difference in the heatmaps became more pronounced. Specifically, the cell closest to the goal exhibited significantly higher value estimates compared to the cell furthest from the goal after extended training. Early on, some cells—especially those further from the goal—displayed near-zero or underdeveloped values, leading to a less defined gradient. However, with additional iterations, the heatmap revealed a more distinct gradient:

- **Cells Near the Goal:** These gradually increased in value as the returns associated with reaching the goal became better sampled and more reliably propagated backward through the state space.
- **Cells Farther from the Goal:** These maintained lower values, but the contrast with the higher-valued cells close to the goal became more apparent as training continued.

The increased gradient difference is indicative of a more refined estimation of long-term expected returns across the state space, emphasizing the distance-related decay in value as the agent moves away from the goal.

Trade-Offs and Comments

Episode Count: Increasing the number of episodes not only helps in reducing the variance inherent to importance sampling but also sharpens the state-value gradient. With more episodes, the differential between the state closest to the goal and the one furthest away becomes more pronounced, reflecting an improved and stabilized value function.

Behavior Policy: A well-designed behavior policy ensures that all parts of the state space are sufficiently explored. I successfully increased convergence speed by emphasizing UP and RIGHT moves, since I have a visual map of the grid and can see those actions have to be taken more often to get to the goal from the starting position.

3.3 On-Policy TD Control (SARSA)

Influence of the Discount Factor (γ) on Value Estimates

Higher γ: The heatmap exhibits smoother gradients as the influence of future rewards is more pronounced, though this was hard to notice since the amount of steps in the episode is short, not allowing the influence of future rewards to be pronounced.

Lower γ: In my observation, this more localized gradient can lead to quicker convergence on the immediate reward structure.

Effect of the Step Size (α) on Convergence Stability

Looking at my graphs, I don't see a very big difference between the two testing parameters (0.2 and 0.1) alone unless combined with an epsilon adjustment. In theory, a larger step size accelerates the updating of Q-values, which can promote faster learning, though it will be more sensitive to changes in the environment and may require more episodes to reach an optimal policy. More testing should be done with varying values of alpha.

Role of the Exploration Rate (ε) in On‑Policy SARSA

Lower ε: A lower exploration rate causes the agent to exploit its current Q-value estimates more heavily, meaning that once the agent finds a policy that gets them to the goal, they tend to exploit it, restricting early exploration in some parts of the state space.

Higher ε: A higher exploration rate encourages a broader search of the state space, which can initially result in a more uniform or "noisy" heatmap. This was largely eliminated with a higher number of iterations, and it was exaggerated with higher alpha.

Heatmap Variations and Final Policy Outcomes

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). The MIT Press.

Early vs. Extended Training: In early iterations, heatmaps can appear patchy, with some cells—particularly those further from the goal—showing near-zero or inconsistent values. With extended training moving from 1,000 to 10,000 episodes, the state values become smoother and the gradient between cells near and far from the goal becomes more pronounced. This indicates that the propagated reward information has stabilized, resulting in a clearer distinction between high-value states (close to the goal) and low-value states (further away).

Optimal Policy and Safety: Importantly, across all hyperparameter settings, I observed that no learned policy ever resulted in fire. This outcome suggests that the on-policy SARSA method, with its inherent conservative nature—updating based on the action actually taken—consistently steers the agent away from dangerous states. The learned policy reliably avoids the fire cell, demonstrating effective risk-averse behavior.

Trade‑Offs and Conclusion

Discount Factor: A higher discount factor promotes long-term planning and a smoother heatmap gradient, but it may require more iterations to settle due to the increased variance from distant rewards. A lower discount factor, by focusing on immediate rewards, leads to a steeper gradient and potentially faster convergence.

Step Size: While a higher step size can boost early learning speed, it risks instability in Q-value estimates, especially when combined with a higher learning rate, according to my observations.

Exploration Rate: Excessive exploration can delay convergence, whereas too little exploration might prematurely lock the policy into suboptimal behaviors, since the agent will tend to keep following a path once it knows it is safe.

3.4 Off-Policy TD Control (Q-Learning)

Influence of the Discount Factor (γ) on Value Estimates

Higher γ: Higher α (0.2) accelerated Q-value updates, promoting rapid learning but causing instability and overshooting optimal values.

Lower γ: When the discount factor is lower, the Q‑Learning update places greater weight on immediate returns. This often results in a steeper gradient around the goal, with more rapid decay of state values for cells far away from the goal. This was evident in cell 10, where it would be darker than its counterpart.

Effect of the Step Size (α) on Convergence

Higher α: A larger step size makes the Q‑Learning updates more aggressive.In some experiments, a high α—especially in conjunction with a higher discount factor and extensive iterations—resulted in a noticeably more negative value in the bottom right cell of the heatmap.

Lower α: A smaller step size yields more conservative updates, which promotes smoother and more stable convergence. As a result, the gradients in the heatmap are more uniformly distributed

Role of the Exploration Rate (ε) in Q‑Learning

Exploration Dynamics: Importantly, across all experiments, no policy resulted in the dangerous "fire" state—demonstrating that the learned policies were not only optimal but also safe.

Heatmap Variations and Final Policy Outcomes

Nicely Distributed Gradients: Across the different hyperparameter settings, the heatmaps show nicely distributed gradients. This was relatively insensitive to hyperparameters, which matches with the theory that we are choosing the max value of each state, requiring less exploration and leading to faster convergence.

Optimal Policy Achievement: Every experiment resulted in an optimal policy, and no policy ever led the agent into the fire (danger) state. I believe this was a result of the updates happening faster, requiring less episodes, leading to a robust policy. More testing should be done, reducing the number of episodes to spot when the policy converges.

Localized Anomalies: In some experiments—particularly those with less iterations, a higher step size, and a higher discount factor—the bottom right cell of the grid was observed to be more negative compared to other regions. This smoothed out over more iterations though.

Trade‑Offs and Comments

Discount Factor: A higher discount factor enhances long-term planning and supports smoother, more globally distributed gradients. However, it can also increase the risk of high variance if not moderated by appropriate step sizes and sufficient episodes.

Step Size: While a larger α can accelerate convergence, it may also induce local anomalies (as seen in the bottom right cell) if the aggressive updates overshoot optimal values in certain regions.

4. Challenges Encountered

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). The MIT Press.

In developing these algorithms, I encountered several challenges with my approach. One of the issues was managing the trade-off between exploration and exploitation across different models. I wished to compare the models using the same hyperparameters, but had to do quite a bit of testing to choose the right set in order to be able to see differences in the comparison. For example, in Monte Carlo methods, the reliance on complete episode returns made them particularly sample inefficient when fewer episodes were available, resulting in sparse value estimates in some regions of the grid. Similarly, with off-policy methods, they tended to converge a lot faster, so choosing the right episode number for a good comparison was difficult.

5. Conclusion and Cross-Comparison of Experiment Results

Looking at the experiments, I can deduce that the effectiveness of the exploration strategy plays a pivotal role in the success of each algorithm. In the on-policy Monte Carlo control experiments, I observed that lower ε values facilitated rapid convergence to a deterministic optimal policy, especially when combined with a moderate discount factor. This configuration encouraged a more focused, exploitative behavior once sufficient state coverage was achieved. However, when a higher exploration rate was used alongside a higher discount factor, the learning process initially appeared more diffuse but ultimately led to a robust, globally informed policy which was evidenced by the smoother gradients in the heatmap.

In the off-policy Monte Carlo experiments, the optimal policy emerges consistently across hyperparameter settings, which was in stark comparison to on-policy MC Control. Since I had a static behaviour policy, I noted that the heatmaps looked roughly identical. This shows how important choosing a well-informed behaviour policy can be.

Turning to the temporal-difference methods, on-policy SARSA demonstrated that smaller step-size updates and lower exploration yielded smoother heatmap gradients, reflecting stable long-term learning while safeguarding against FIRE state risk. Conversely, off-policy Q-Learning TD control uses an aggressive update strategy though the maximum future Q-value, which resulted in faster convergence and less sensitivity to hyperparameters.

Overall, my experiments indicate that while each algorithm has its strengths and vulnerabilities, Q-Learnig's ε-greedy exploration strategy consistently underpins successful policy convergence. I was delightfully surprised by the off-policy MonteCarlo control using my behaviour policy, and how consistent the heatmaps looked, proving to result in smooth heatmap gradients.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). The MIT Press.