CS 138: Homework 3

Lawrence Qiu

November 5, 2025

1 Introduction

Exercise 7.10 of Sutton and Barto asks us to devise an off-policy prediction problem and compare the per-decision G calculation method with the standard direct ρ weighting method, showing that the per-decision method is more data-efficient than the direct weighting method.

In off-policy control problems, an exploratory policy b is used to interact with the environment and generate samples, but the calculated value or Q function corresponds to a different policy π (often the greedy policy). One method to adapt an on-policy method for an off-policy problem is to use importance sampling. In importance sampling, updates to the value function are weighted by the relative likelihood of generating the trajectory under the target policy versus under the exploratory policy. This can be extended to n-step TD by multiplying the relative probabilities for each step:

$$\rho_{t:h} \doteq \prod_{k=t}^{\min(h,T-1)} \frac{\pi(A_k \mid S_k)}{b(A_k \mid S_k)},$$

with the value update function

$$V_{t+n}(S_t) \doteq V_{t+n-1}(S_t) + \alpha\rho_{t:t+n-1}\left[G_{t:t+n} - V_{t+n-1}(S_t)\right].$$

The disadvantage of this method is that if on any step of the n-step TD trajectory the probability of the target policy is zero, then the entire relative likelihood becomes zero, causing the update to be zero. This issue is reasonable: given an ϵ-greedy exploratory policy, there is an ϵ likelihood of choosing the non-optimal action (assuming a large action count), and the cumulative likelihood of choosing any non-optimal action increases exponentially with n.

An alternative is the per-decision method. Rather than correcting for bias with the whole trajectory, the bias is corrected for on each step of the n-step trajectory by bootstrapping unlikely target policy actions with the existing value function (known as the control variate):

$$G_{t:h} \doteq \rho_t\left(R_{t+1} + \gamma G_{t+1:h}\right) + (1-\rho_t)V_{h-1}(S_t).$$

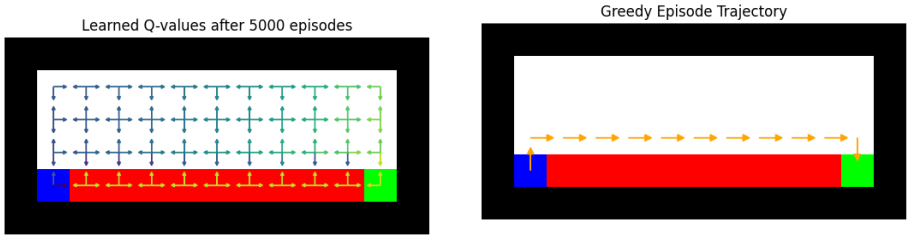
Then, the final value update function can be identical to the non-importance sampling update function:

$$V_{t+n}(S_t) \doteq V_{t+n-1}(S_t) + \alpha \left[G_{t:t+n} - V_{t+n-1}(S_t) \right].$$

2 Part 1: Implementing the Exercise

For the environment, I recreated the cliff navigation problem where the agent is given -1 reward on each step and a -10 reward if it falls off the cliff. The agent can either go right next to the cliff (reducing path length) or go around and reduce the likelihood of falling down if the policy isn't optimal. This was to verify if my implementation of importance sampling is correct, as the expected final Q-function should match that of the optimal Q-function (where the greedy path is right next to the cliff) even when the behavioral policy is ϵ-greedy.

The control method I used is n-step Sarsa with importance sampling, using an ϵ-greedy behavioral policy and a greedy target policy. For the following experiments, I used the parameters $\epsilon = 0.1$, $\gamma = 0.9$, $\alpha = 0.01$, and $n = 3$, and simulated for 5000 episodes.



(a) Q values after learning with off-policy n-step Sarsa.

(b) Greedy trajectory using learned Q-values, with path next to cliff indicating correct implementation.

Figure 1: Results of off-policy n-step Sarsa on the cliff environment.

To evaluate the significance of the "zeroed out" importance sampling ratio issue, I computed the percentage of updates that were discarded per episode, shown in Figure 2. The discarded ratio stands at about 20%, which is close to the calculated amount: $1 - ((1 - \frac{3}{4}\epsilon))^n \approx 20.8\%$.

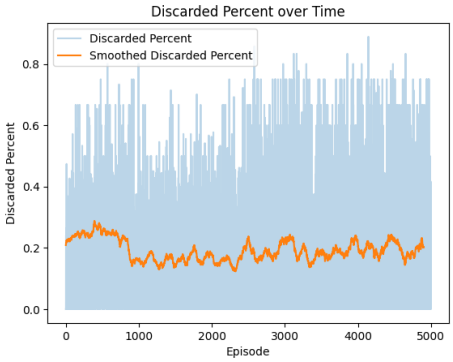
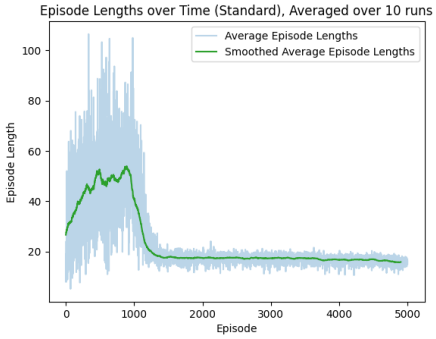


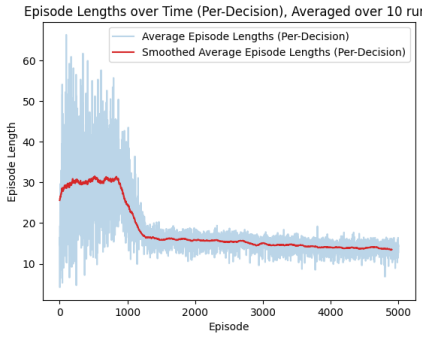
Figure 2: Percent of zero-ed out TD updates by episode.

Next, I implemented a modified version of off-policy Sarsa with per-decision G calculation. This is identical to the value function-based G calculation method mentioned previously except that the importance sampling calculation is not done for the first action in the n-step TD update trajectory. With the same parameters as before, the resulting learned Q-values are identical, indicating correct implementation.

Then, I compared the sample efficiency between the two methods by averaging the by-episode episode length over 10 runs of each method.



(a) By-episode episode lengths averaged over 10 runs using standard off-policy Sarsa.



(b) By-episode episode lengths averaged over 10 runs using per-decision off-policy Sarsa.

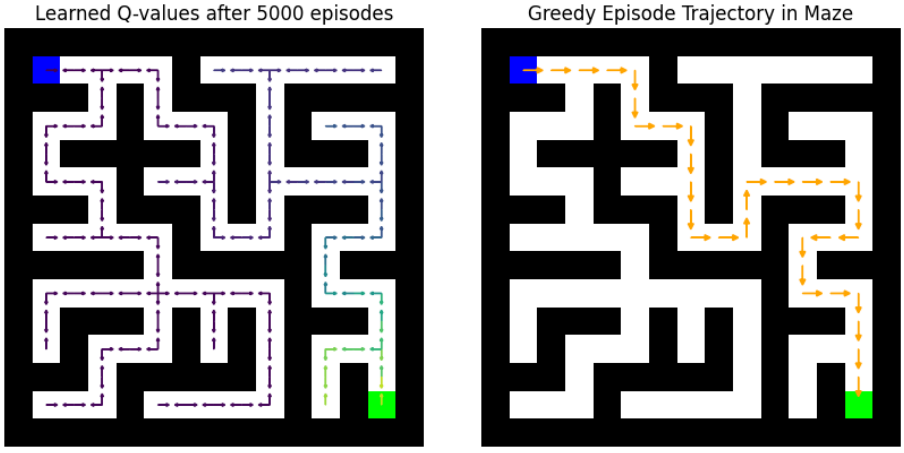
Figure 3: Comparison of learning efficiency between standard and per-decision off-policy Sarsa.

Interestingly, the shape of the standard method shows a "hump" at the beginning of the run that doesn't appear in the per-decision method. Averaging the episode length across episodes, the final average episode length for the standard method is 23.43 whereas the average episode length for the per-decision

method is 18.20, indicating a significant decrease in total length meaning higher sample efficiency.

3 Part 2: Alternative Maze Environment

To explore the benefits of the per-decision update method over the standard method more, I devised a different maze environment, in which the number of steps between the starting and end states would be higher. This would benefit more from a higher n in n-step Sarsa, and a higher n would negatively affect the standard method more (as the likelihood of zero-ed out TD updates increases exponentially with n) compared to the per-decision method.



(a) Learned Q-values after 5000 episodes on maze environment.

(b) Greedy path using previously learned Q-values.

Figure 4: Results of standard off-policy n-step Sarsa on the maze environment.

Then, I performed a hyperparameter search over n, with n ranging from 1 to 20 on a roughly logarithmic interval. I averaged the average episode length over 10 runs and plotted that against n for both the standard and per-decision methods. For low n values ($n < 5$), the performance of the two methods is about the same, but for higher n values, the per-decision method remains stable while the standard method performance decreases due to zero-ed out TD updates.

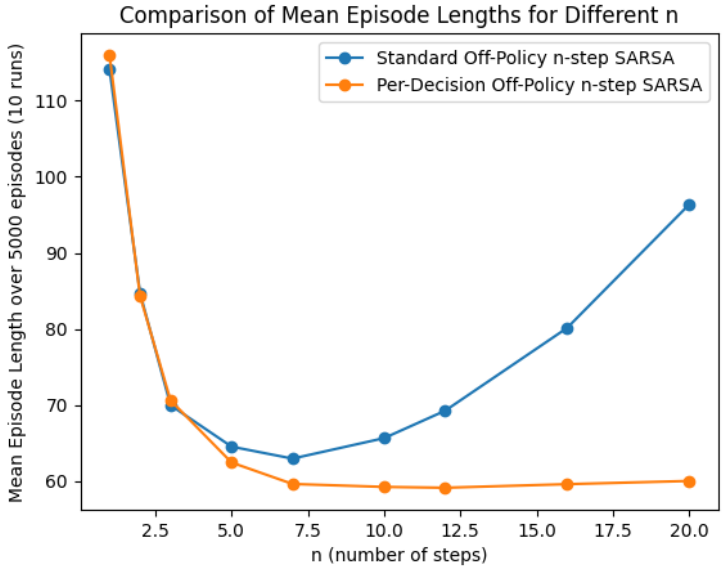


Figure 5: Average episode length over 10 runs vs. value of n for standard off-policy n-step Sarsa and per-decision off-policy n-step Sarsa.

4 Summary and Conclusion

I implemented importance-sampling-based off-policy Sarsa and showed the implementation is correct on the cliff-walking environment. Then, I implemented the per-decision method where unlikely target policy actions are bootstrapped on each step of the TD-update trajectory with the existing value function. The average episode length using the new method is significantly lower than the standard method. Next, I created a maze environment with higher step count and compared the performance of both methods versus n. As expected, for low n both methods are similar, but as n increases, the performance of the standard method rapidly decreases while it remains stable for the per-decision method.