Reinforcement Learning CW1

s2704516

March 24, 2025

1 Question 5

In this question, I compared SARSA and Q-learning in the FrozenLake environment, building on the base methods from Question 2. The code was adapted to utilize the *Run* class, allowing execution with multiple seeds to ensure reproducibility and generalization. I modified the original Q-learning implementation to incorporate SARSA, facilitating a direct comparison by training both methods in the same environment. Various metrics were recorded for analysis while keeping all other hyperparameters unchanged to ensure a fair comparison. Additionally, I experimented with different parameters of the linear decay presented in the exercise; a less aggresive version. This approach adjusts the exploration rate from 1.0 to 0.05 over the first 50% of training steps. Initially, actions are mostly random, encouraging exploration. As training progresses, the strategy shifts towards exploitation, stabilizing epsilon at 0.05 for consistent exploration. This balance helps prevent premature convergence and ensures thorough policy learning. Both strategies were tested using the slippery variant of the problem to examine how each method handles stochasticity.

The code and data saved can be found under the directory of *exercise 5*.

1.1 FrozenLake 8×8 Environment

FrozenLake 8×8 is a reinforcement learning environment where an agent navigates a grid-based frozen lake to reach a goal while avoiding holes. The environment has two variants; a non-slippery variant where the agent's actions deterministically move it in the intended direction; and a slippery variant where the agent's movements are stochastic, meaning actions have a probability of leading to unintended directions, increasing uncertainty and making optimal planning more challenging.

1.2 SARSA vs. Q-Learning

Both SARSA and Q-learning are Temporal Difference (TD) learning methods, meaning they update value estimates based on bootstrapping rather than waiting for full episodes, like Monte Carlo methods.

1.2.1 Q-learning (Off-policy)

Q-learning is an off-policy method updates its Q-values using the **maximum possible future reward**, independent of the agent's actual policy:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{1}$$

This approach makes Q-learning more **exploitative** and effective in deterministic environments as it can lead to faster learning optimal policies, but can be unstable in stochastic settings, such as the slippery variant of FrozenLake.

1.2.2 SARSA (On-policy)

SARSA, in contrast, is an on-policy method, that updates Q-values based on the **actual action taken** by the agent following its policy:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma Q(s',a') - Q(s,a) \right] \tag{2}$$

Since SARSA follows the agent's behavior policy, it results in **safer learning** in stochastic environments but may converge to a suboptimal policy if exploration is not well-balanced.

The key difference is that **Q-learning is off-policy** (optimistically updates based on the best possible future action), making it more **aggressive and exploitative**, while **SARSA is on-policy** (follows the agent's behavior), leading to **more cautious, stable learning**, especially in slippery environments like FrozenLake.

1.3 Exploration strategies

The two exploration strategies primarily differ in the rate and duration over which the exploration parameter, epsilon, decays. The first strategy features a rapid decay, reducing epsilon from 1.0 to 0.01 within just 20% of the total training steps. This swift reduction forces a quick transition from exploration to exploitation, suitable for simpler environments where less exploration is needed to find an optimal policy. The second strategy, however, applies a more gradual decay, lowering epsilon from 1.0 to 0.05 over 50% of the training steps. This slower approach allows for more extensive exploration, which can be crucial in complex environments where the agent benefits from exploring a wider range of actions and states before committing to an exploitation phase.

2 Results

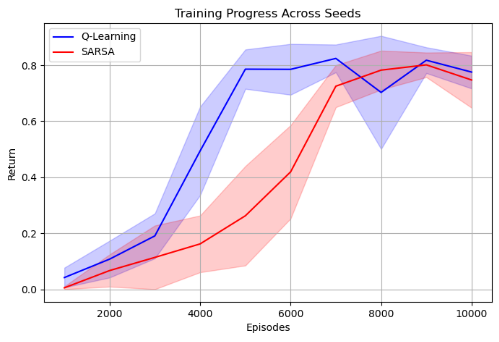


Figure 1: Original Linear Decay

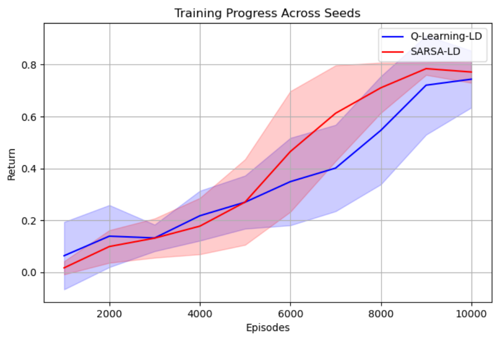
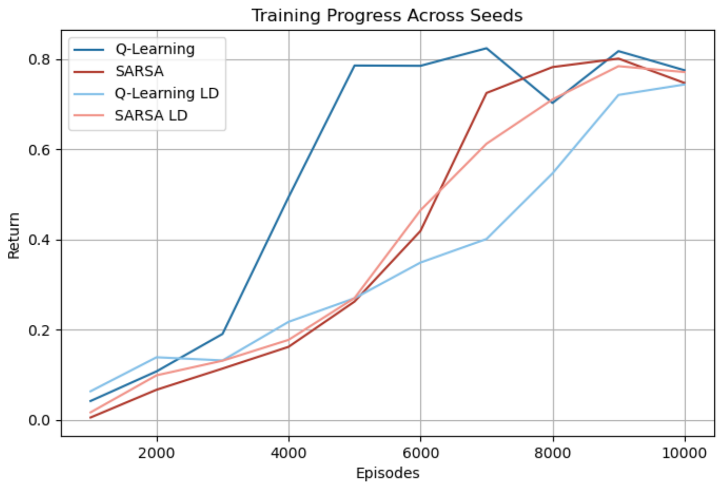


Figure 2: Implemented Linear Decay

The two plots compare the performance of Q-learning and SARSA under different exploration strategies in the slippery FrozenLake environment. The left plot uses a more aggressive epsilon decay, where exploration drops rapidly within 20% of training. Here, Q-learning clearly outperforms SARSA, achieving higher returns earlier and converging faster, though with more variance. At the end, they do converge to the same values. SARSA lags behind, likely due to insufficient exploration during its early learning phase, as the slope is definitely smaller, which is critical in a stochastic setting. In contrast, the right plot uses the slower linear decay I implemented, where epsilon decays from 1.0 to 0.05 over 50% of training. This strategy yields a narrower performance gap between the two algorithms. SARSA, in particular, benefits significantly from the extended exploration phase, catching up and even outperforming Q-learning in later episodes. This suggests that SARSA, being on-policy, relies more heavily on sufficient exploration to build stable estimates, while Q-learning, as an off-policy method, is more robust to early exploitation. Overall, the comparison highlights that exploration strategy can significantly influence relative algorithm performance, particularly in environments with stochastic dynamics. This is very important in computationally harder environments as we want the fastest convergence.

3 Conclusion

The plot compares Q-learning and SARSA under two exploration strategies in stochastic environments: the original fast-decay and the slower linear decay (LD) version I implemented. Q-learning with the original aggressive decay achieves the highest performance the fastest, converging early and maintaining strong returns but being very variable. SARSA initially lags behind but eventually catches up after around 8,000 episodes and they converge to a similar value. When using the slower decay (LD), both algorithms show more gradual progress. Interestingly, SARSA benefits significantly from the longer exploration phase, slightly surpassing its original version at the end. In contrast, Q-learning LD converges more slowly and underperforms compared to its aggressive-decay counterpart and all the other methods. This suggests that while Q-learning can thrive with fast exploration reduction, SARSA depends more on extended exploration to perform well. This difference can be attributed to their inherent differences. Overall, the slower linear decay improves SARSA's performance and stability, while Q-learning performs best when allowed to exploit earlier.


Training Progress Across Seeds
Q-Learning
SARSA
Q-Learning LD
SARSA LD
Return
Episodes