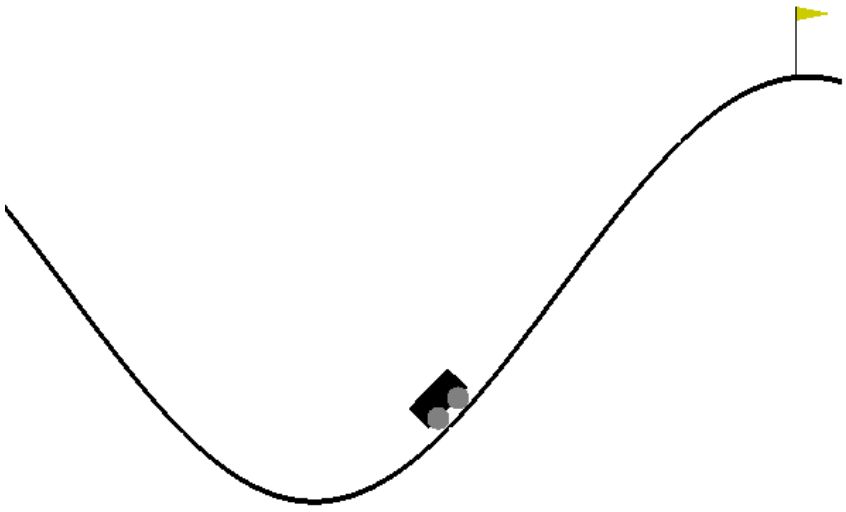
Dauphine | PSL
UNIVERSITÉ PARIS

Reinforcement Learning
Experimenting the impact of stochasticity and policy choice on TD(0) Agents behavior

Pierre Neveu and Salma Lahrach

January 2025



Contents

1 Introduction

The objective of this project is to implement **Time Difference (TD) learning agents** in environments of different complexities. We took advantage of this project to learn how to design a code architecture that ensures flexibility in the context of implementing reinforcement learning agents. Therefore, we aimed for the most **modular implementation** possible and it allowed us for dynamic integration of new agents, policies, environments, and experimental setups. A class diagram is given here: A.

Initially, we focused on **Q-learning** and **SARSA** for *Mountain Car*. An interpretation of these algorithms would be that Q-Learning, as an off-policy one, is not realistic / too optimistic because we cannot ensure that he learns what would happen in reality.

However, we quickly observed that in Mountain Car, their value updates are very similar. This can be attributed to the use of an ε**-greedy policy** in such environment as they require a small ε, which results in similar action selection and q-values updates across both algorithms (if $\varepsilon = 0.01$, SARSA will select the argmax action 99% of the time and both algorithms will have a very similar behavior). Moreover, *Mountain Car* is a **deterministic environment**. All of this preliminary observations have led us to ask ourselves:

1. *What is the impact of the stochasticity of the environment on the behavior of SARSA and Q-Learning ?*

2. *How does the policy used influence their behavior ?*

The set of possible experiments that could answer these questions is very large so we will focus on comparing ε**-greedy policy** and **softmax policy**. As Mountain Car doesn't allow much the randomness to be varied, we will use the Cliff Walking environment and modify its implementation to vary the slip factor. We optimized our agents on both environments under both policies. Then we keep the same agent for every experiments in order to have a fair comparison.
We will first present our results of their behavior in the complex but mostly deterministic environment of Mountain Car.
With the Cliff Walking environment, we will then try to identify to what extent the stochasticity of the actions influence them.

2 Mountain Car

Gym doc: www.gymlibrary.dev/environments/classic$_c ontrol/mountain_c ar/$
Code used for state discretization: http://incompleteideas.net/tiles/tiles3.html

The only randomness that we can control is the starting state, which lies in an interval. By default, the starting state is random. After presenting how the agents are tuned in the default settings, we will try to make the starting state deterministic to evaluate the impact of such a change.

2.1 Tuning SARSA and Q-learning Agents

We will present only the SARSA tuning, but Q-Learning is similar.

2.1.1 Stage 1: Comparison of GLEI and Standard ε-Greedy

During initial experiments with SARSA, it was not obvious whether using a GLEI strategy offered any significant advantage over a standard ε-greedy policy. Therefore, the first grid search was conducted.

Experimental Setup:

- Learning rate α was fixed at 0.1, and discount factor γ was set to 0.99.
- The following $\varepsilon = [0.005, 0.01, 0.05, 0.1]$
- For each value of ε, experiments were run with and without the GLEI strategy.
- Trained on 1000 episodes.

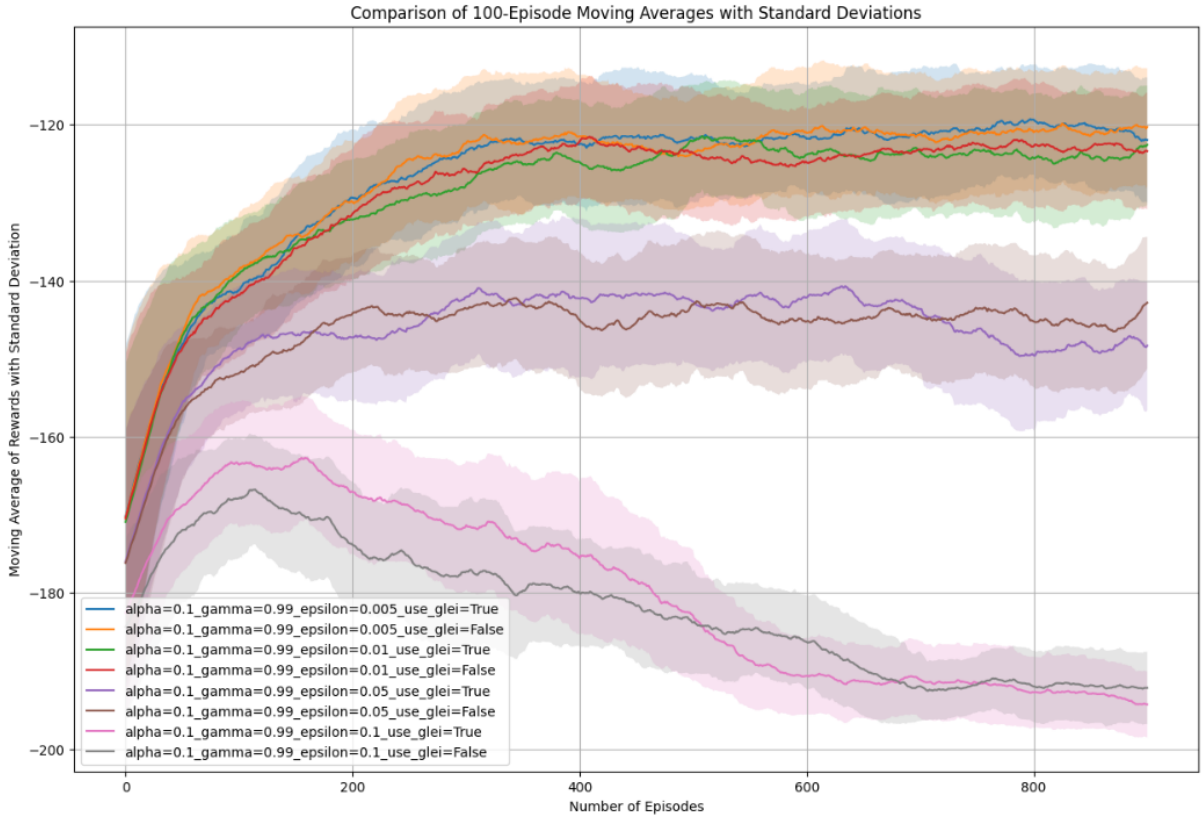


Figure 1: SARSA training curves on Mountain Car environment for different ε

The overall results show that the GLEI strategy offers no significant advantage in terms of optimality or convergence speed. For $\varepsilon = 0.005$ and $\varepsilon = 0.01$, the results are ambiguous. For $\varepsilon = 0.005$, the standard training curve ends better. For $\varepsilon = 0.01$, the GLEI training curve ends better. Even if the GLEI strategy does not necessarily give significant advantages, we must keep in mind that it theoretically ensures that SARSA and Q-Learning converge asymptotically to the optimal policy.

2.1.2 Stage 2: Learning rate and discount factor optimization

With the GLEI strategy and starting ε fixed, we conducted a second grid search to tune the learning rate α and discount factor γ.

Experimental Setup:

- $\alpha = [0.005, 0.01, 0.05, 0.07, 0.1]$
- $\gamma = [0.9, 1]$
- $\varepsilon = 0.005$
- Use GLEI strategy.
- Trained on 1000 episodes

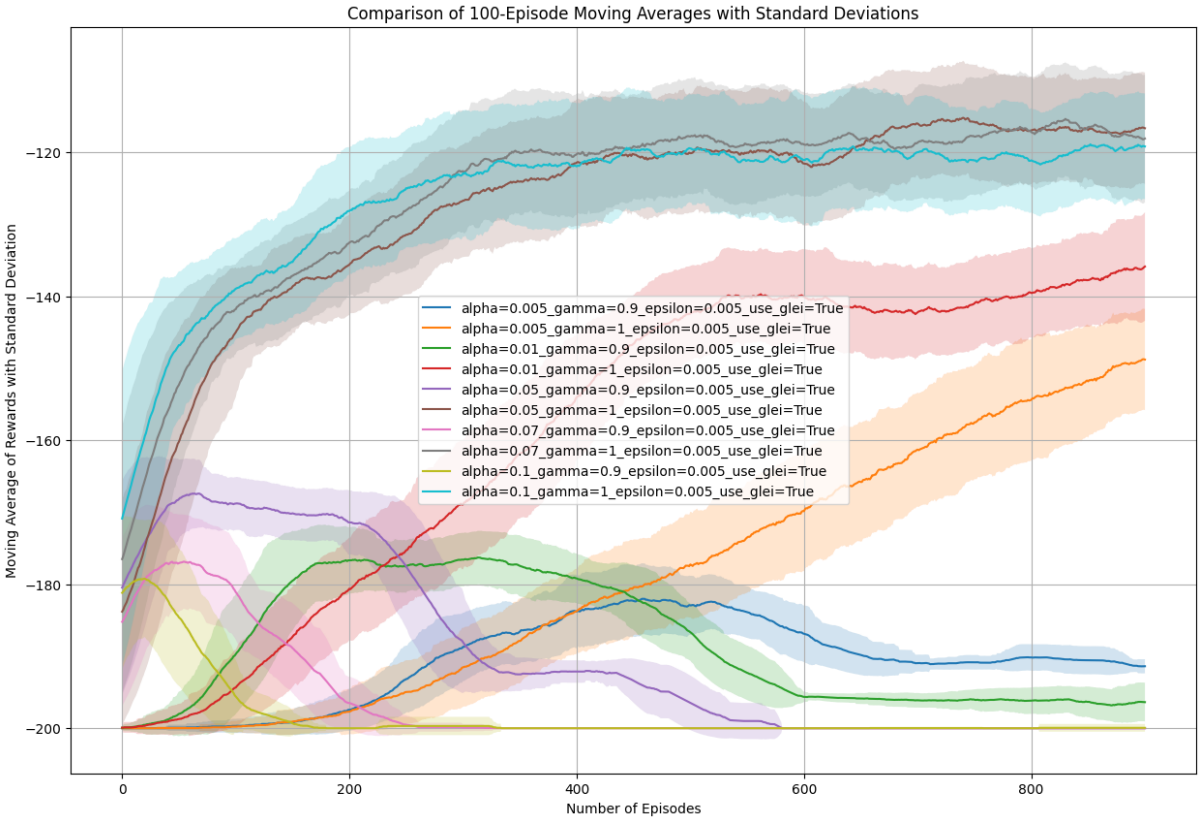


Figure 2: SARSA training curves on Mountain Car environment for different α and γ

2.2 Comparison of Fine-Tuned Models

In this section, we compare the performance of the fine-tuned SARSA and Q-learning models on the *Mountain Car* environment using ε-Greedy Policies. To gain a deeper understanding of their behaviors, we evaluated the policies at regular intervals during training. The final evaluation corresponds to the policy learned at the end of the training process.

Experimental Setup:

- One training is made on 3000 episodes
- The agent is evaluated 4 times during one training.
- An evaluation at episode x consists of the mean score over 10 episodes with a hard policy.
- We perform 10 trainings for each agent.

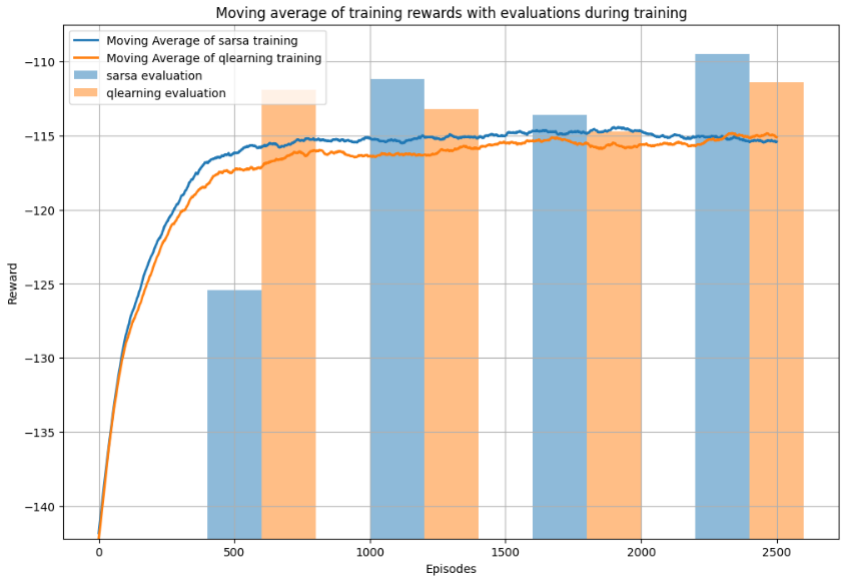


Figure 3: Comparison of Training and Policy Evaluation for Fine-Tuned SARSA and Q-learning Models on Mountain Car

The results, shown in Figure 3, align well with theoretical expectations: they both converge towards the same policy (that should be optimal). However, theory doesn't guarantee how they will converge to this optimal policy: they may converge before to sub optimal policies. What we see here confirms our first assumption according to which **ε-greedy policy** with a very deterministic environment will result in the two algorithms behaving in very similar way. The true difference here is that q-learning converges faster but experiments shows it may not be systematically the case.

2.3 Determinist start in Mountain Car

We fix the seed for starting state (`seed=42`) to ensure the same initial state for every episode. For evaluation, we don't fix the seed so the initial state becomes again stochastic.

Without any surprise, this setting does not allow the agent to explore enough. Therefore, we don't use glei strategy and fix epsilon and temperature to ensure that agents learn. We observe that using a softmax policy gives a better and more stable learning to the algorithms. In both situations, SARSA looks to have better results in evaluation, it might be due to a better exploitation of its policy and the exploration it implies.

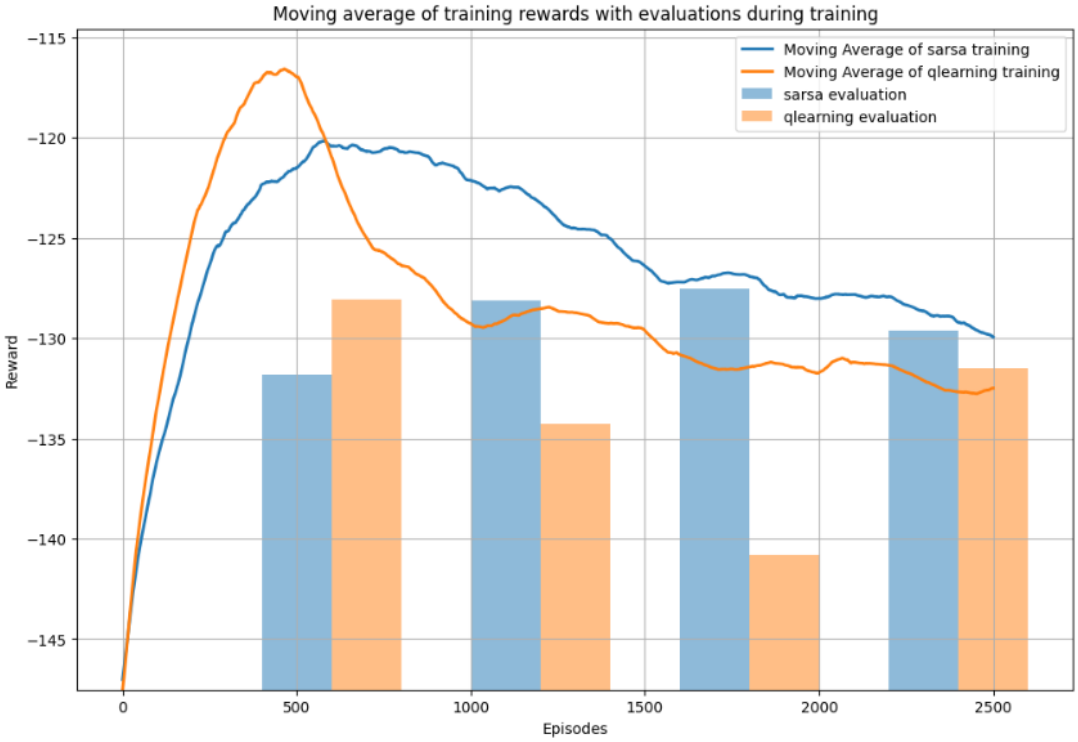


Figure 4: Comparison Q-Learning with on Mountain Car with determinist start and ε-**greedy policy**

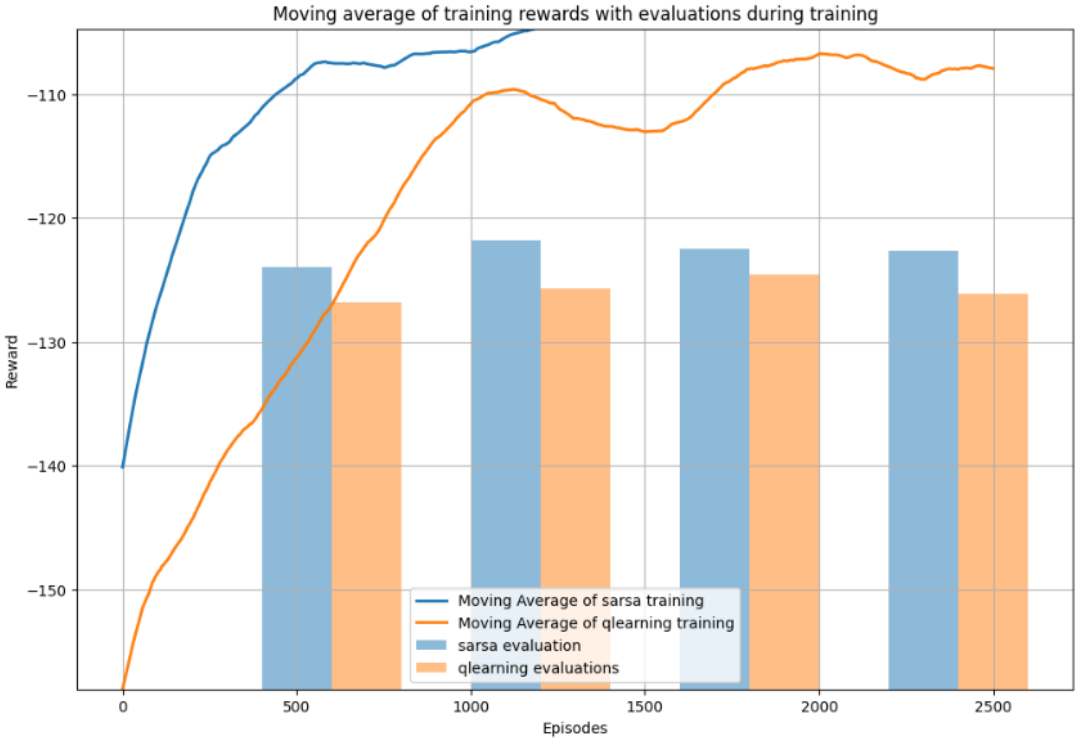


Figure 5: Comparison Q-Learning with on Mountain Car with determinist start and **softmax policy**

3 Cliff Walking

Gym doc: https://gymnasium.farama.org/environments/toy$_text/cliff_walking/$

3.1 Tuning agents

We applied the same tuning methodology used in the *Mountain Car* environment to fine-tune SARSA and Q-learning agents in *Cliff Walking*. A interesting change is that the optimal discount is now 0.8. It probably leads the agent to be less punished from falling in the cliff and therefore learn faster the optimal policy.

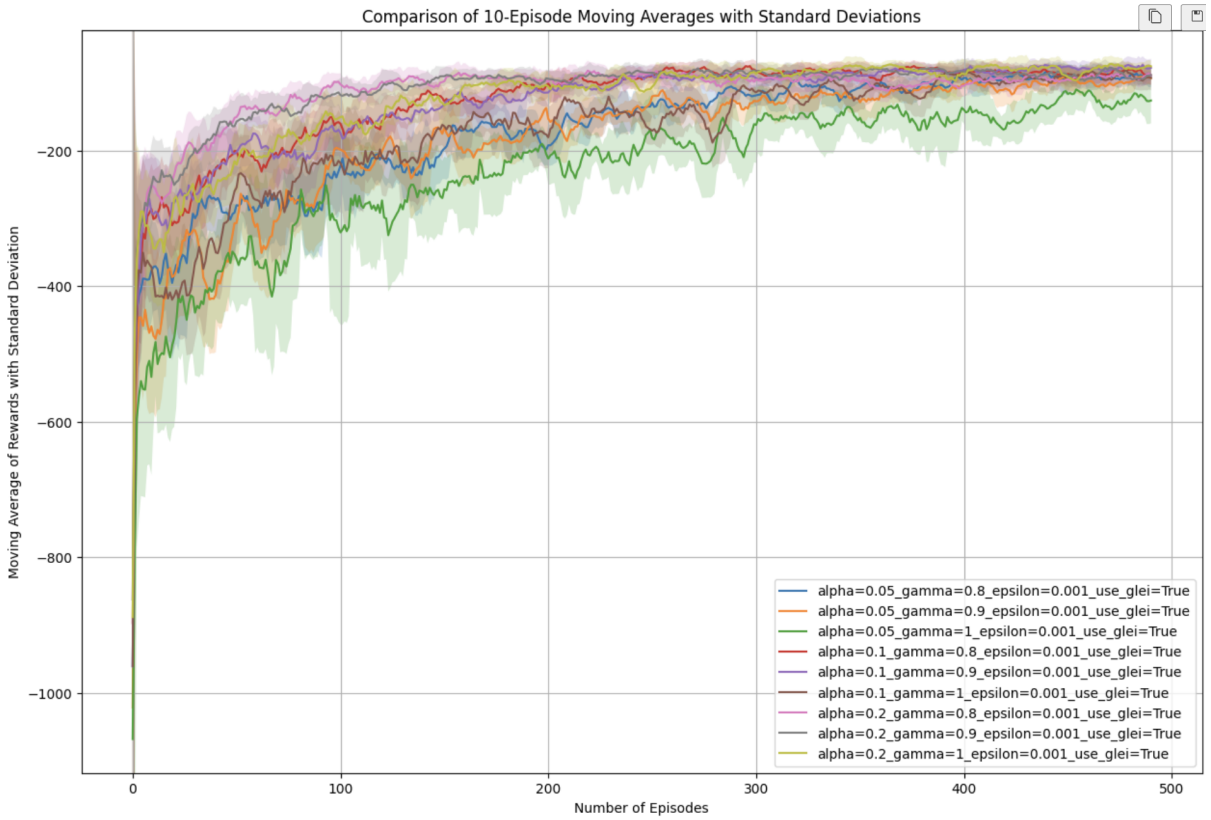


Figure 6: Tuning SARSA with ε-greedy policy on Cliff Walking environment

3.2 Customizing Stochasticity in Cliff Walking

In order to analyze the impact of stochasticity, we implemented a customized version of the *Cliff Walking* environment, which allowed us to vary the `slippery_prob` parameter. The source code for this customized environment is provided in **Appendix A.2**.

Experimental Setup:

- We used the parameter tuned in the default environment (slippery prob. $= 0.33$). See previous section.
- The parameter `slippery_prob` was tested with several values in the range $[0, 1]$:

3.3 Performance with Different Levels of Stochasticity

3.3.1 Case 1: Deterministic Environment, slippery prob. = 0

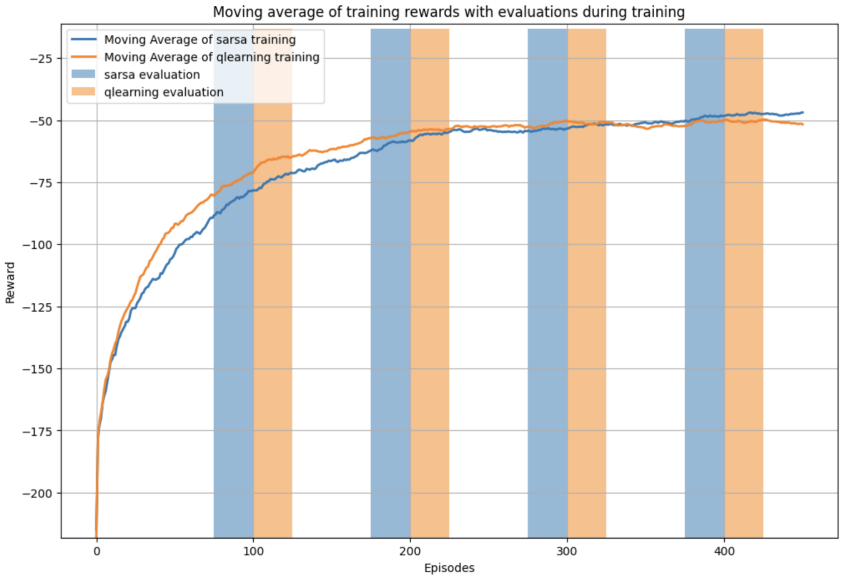


Figure 7: Comparison Q-Learning with ε-greedy policy on Cliff Walking environment with slippery prob = 0

3.3.2 Case 2: Moderate Stochasticity, slippery prob. = 0.33

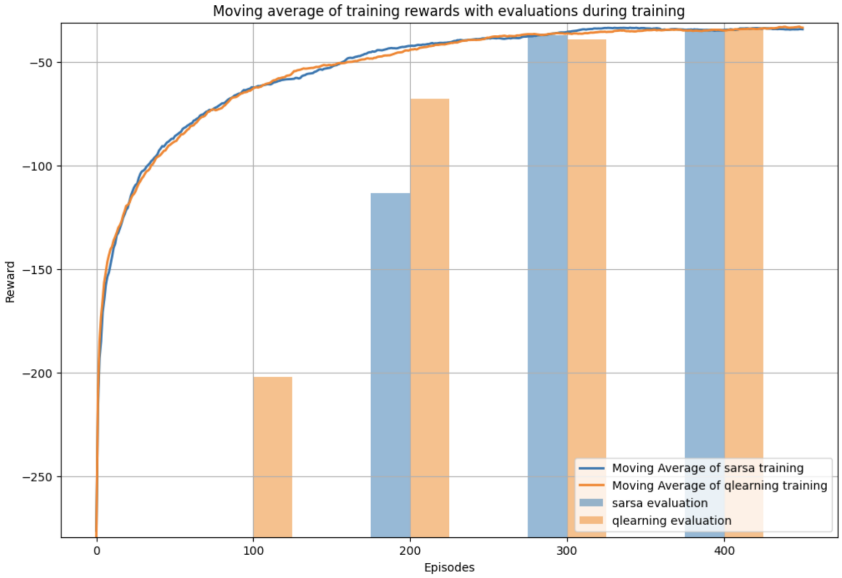


Figure 8: Comparison Q-Learning with ε-greedy policy on Cliff Walking environment with slippery prob = 0.33

3.3.3 Case 3: Extreme Stochasticity, slippery prob. = 1

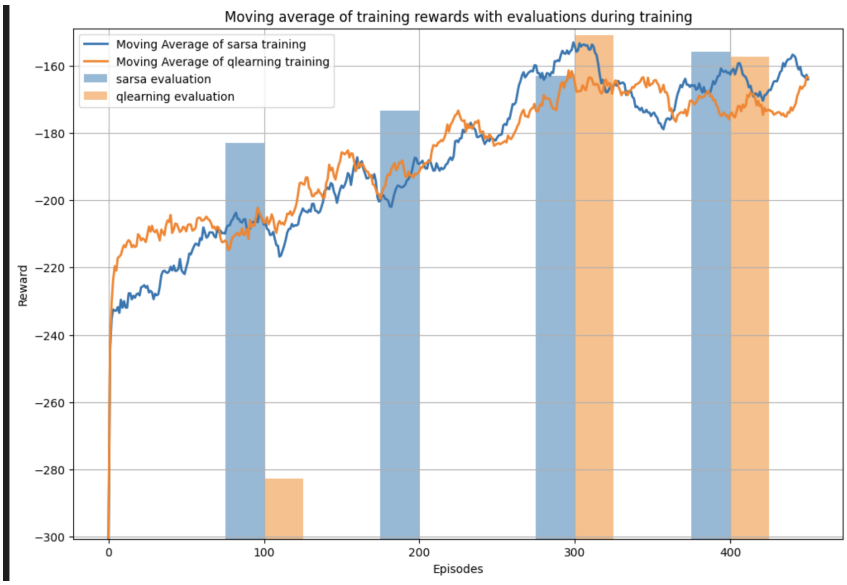


Figure 9: Comparison Q-Learning with ε-greedy policy on Cliff Walking environment with slippery prob = 1

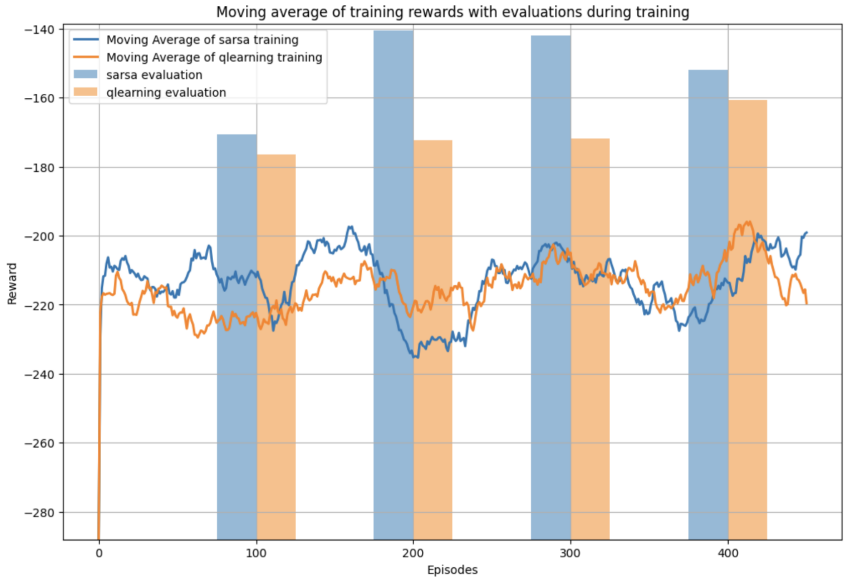


Figure 10: Comparison Q-Learning with softmax policy on Cliff Walking environment with slippery prob = 1

4 Conclusion

This project provided practical insights into the behavior of SARSA and Q-learning under ε-greedy policy and softmax policy in environments of different complexities and stochasticity. Regarding the results in our context, the main conclusions are:

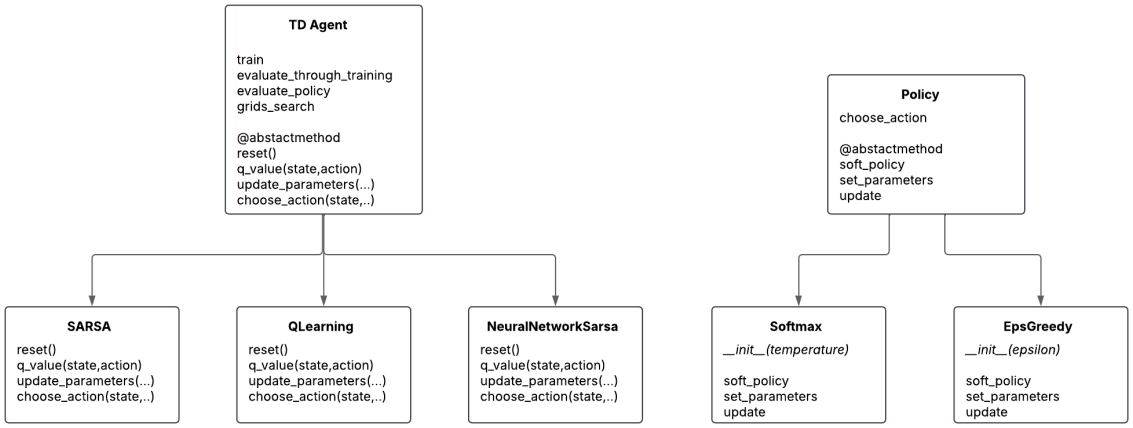
1. Regardless of the policy used, a fully determinist environment leads both algorithms to behave almost identically except that Q-Learning may converge slightly faster when environment becomes more complex (See figures. **Figures 3 and 7, 8**).

2. A fully stochastic environment leads both algorithms to behave differently in training but still converge to the same policy under GLEI strategy just like the theory states. In theses conditions, Q-Learning has worst rewards in training.(See figures. **Figures 9**).

3. Even though it converges toward "a surrounding" of the optimal policy, softmax policy offers a more stable training in a fully stochastic environment and ensure that the agent explores enough in setting where exploration is disadvantaged.(See figures. **Figures 5 and 10**).

A Appendix

A.1 Class Diagram



A.2 Appendix: Customized Cliff Walking Environment

The following code implements a customized version of the *Cliff Walking* environment, allowing the `slippery_prob` parameter to be varied:

```
UP = 0
RIGHT = 1
DOWN = 2
LEFT = 3

class CustomCliffWalkingEnv(CliffWalkingEnv):
    """
    A custom Cliff Waling environment from gymnasium.

    This environment is used to study the impact of different probabilities of slippery.
    The easier way to proceed to a custom probability is to set slippery of the initial environment to False. Then,
    overwritte the step method and take a perpendicular action with the given probability (self.p)
    """

    def __init__(self, render_mode=None, is_slippery=False, slippery_prob=0.33):
        super().__init__(render_mode=render_mode, is_slippery=is_slippery)
        self.slippery_prob = slippery_prob  # Default probability for random actions

        # Proba to actually take the selected action.
        no_slippery_prob = 1 - self.slippery_prob

        # Probability of perpendicular actions to be taken.
        side_slippery_prob = self.slippery_prob / 2

        # Keys: action, values: array of probability of slippery to action each other action (actions are the index of array)
        self.transition_prob = {
            UP: [no_slippery_prob, side_slippery_prob, 0, side_slippery_prob],
            RIGHT: [side_slippery_prob, no_slippery_prob, side_slippery_prob, 0],
            DOWN: [0, side_slippery_prob, no_slippery_prob, side_slippery_prob],
            LEFT: [side_slippery_prob, 0, side_slippery_prob, no_slippery_prob],
        }

```

```
    def reset(self, seed=None):
        state, _ = super().reset(seed=seed)

        return state, {"slippey_prob": self.slippery_prob}

    def step(self, action):
        # Call the parent step method
        state, reward, done, truncated, _ = super().step(
            np.random.choice(
                a=[UP, RIGHT, DOWN, LEFT], p=self.transition_prob[action]
            )  # Choosing the action according to the probability of slippery
        )

        return state, reward, done, truncated, {"slippey_prob": self.slippery_prob}
```

References

[Singh et al.(2000)Singh, Jaakkola, and Littman] Satinder Singh, Tommi Jaakkola, and Michael L. Littman. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38:287–308, 2000.

[Sutton and Barto(2018)] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018.