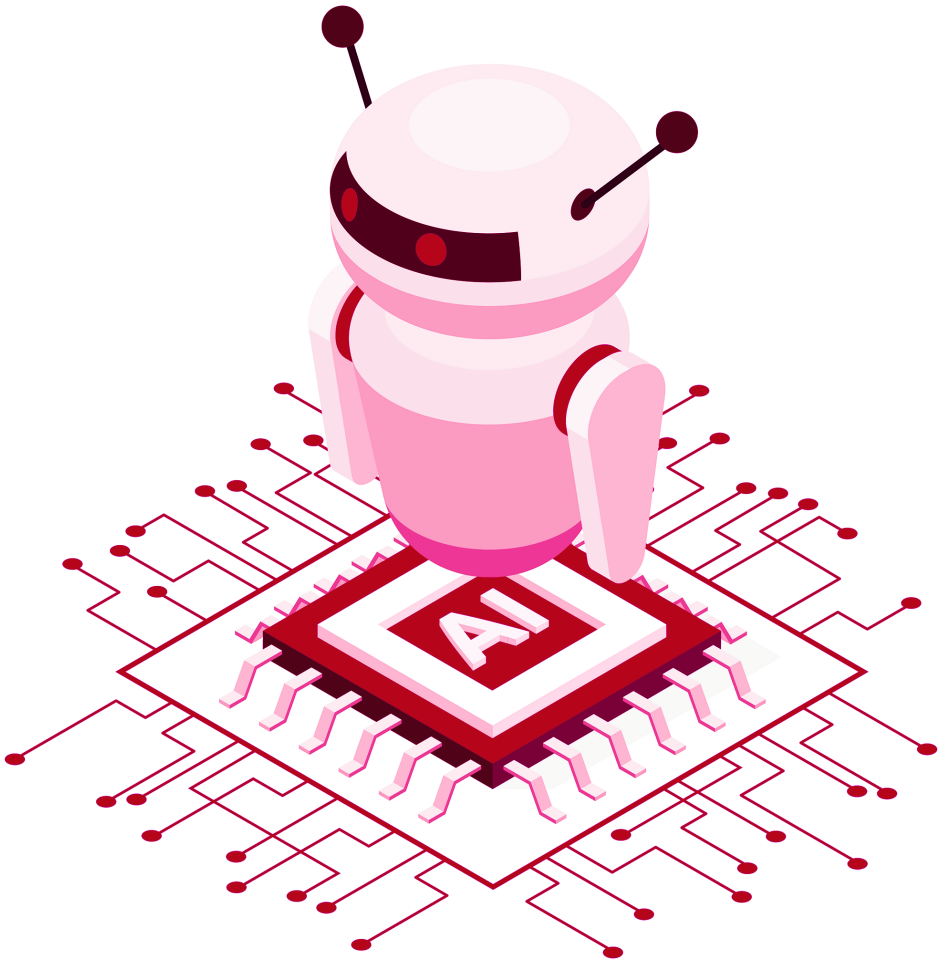


Deep Reinforcement Learning

Professor Mohammad Hossein Rohban

Homework 2:

Value-Based Methods

By:

Ayeen Poostforoushan

401105742

Spring 2025

Contents

1 Epsilon Greedy **1**

1.1 Epsilon 0.1 initially has a high regret rate but decreases quickly. Why is that? [2.5-points] 1

1.2 Both epsilon 0.1 and 0.5 show jumps. What is the reason for this? [2.5-points] 1

1.3 Epsilon 0.9 changes linearly. Why? [2.5-points] 1

1.4 Compare the policy for epsilon values 0.1 and 0.9. How do they differ, and why do they look different? [2.5-points] 1

1.5 In the epsilon decay section, analyze the optimal policy for the row adjacent to the cliff (the lowest row). Then, compare the different learned policies and their corresponding rewards. [2.5-points] 2

2 N-step Sarsa and N-step Q-learning **2**

2.1 What is the difference between Q-learning and sarsa? [2.5-points] 2

2.2 Compare how different values of n affect each algorithm's performance separately. [2.5-points] 3

2.3 Is a Higher or Lower n Always Better? Explain the advantages and disadvantages of both low and high n values. [2.5-points] 3

3 DQN vs. DDQN **4**

3.1 Which algorithm performs better and why? [3-points] 4

3.2 Which algorithm has a tighter upper and lower bound for rewards. [2-points] 4

3.3 Based on your previous answer, can we conclude that this algorithm exhibits greater stability in learning? Explain your reasoning. [2-points] 4

3.4 What are the general issues with DQN? [2-points] 5

3.5 How can some of these issues be mitigated? (You may refer to external sources such as research papers and blog posts be sure to cite them properly.) [3-points] 5

3.6 Based on the plotted values in the notebook, can the main purpose of DDQN be observed in the results? [2-points] 5

3.7 The DDQN paper states that different environments influence the algorithm in various ways. Explain these characteristics (e.g., complexity, dynamics of the environment) and their impact on DDQN's performance. Then, compare them to the CartPole environment. Does CartPole exhibit these characteristics or not? [4-points] 5

3.8 How do you think DQN can be further improved? (This question is for your own analysis, but you may refer to external sources such as research papers and blog posts be sure to cite them properly.) [2-points] 6

Grading

The grading will be based on the following criteria, with a total of 100 points:

Task	Points
Task 1: Epsilon Greedy & N-step Sarsa/Q-learning	40
Jupyter Notebook	25
Analysis and Deduction	15
Task 2: DQN vs. DDQN	50
Jupyter Notebook	30
Analysis and Deduction	20
Clarity and Quality of Code	5
Clarity and Quality of Report	5
Bonus 1: Writing your report in Latex	10

Notes:

- Include well-commented code and relevant plots in your notebook.
- Clearly present all comparisons and analyses in your report.
- Ensure reproducibility by specifying all dependencies and configurations.

1 Epsilon Greedy

1.1 Epsilon 0.1 initially has a high regret rate but decreases quickly. Why is that? [2.5-points]

The initial policy is random/arbitrary. So it doesn't work well and because of the small epsilon, it takes some time to explore the good trajectories. So initially we have a high regret rate but when we explore the good direction, we exploit it and have low regret rate after.

1.2 Both epsilon 0.1 and 0.5 show jumps. What is the reason for this? [2.5-points]

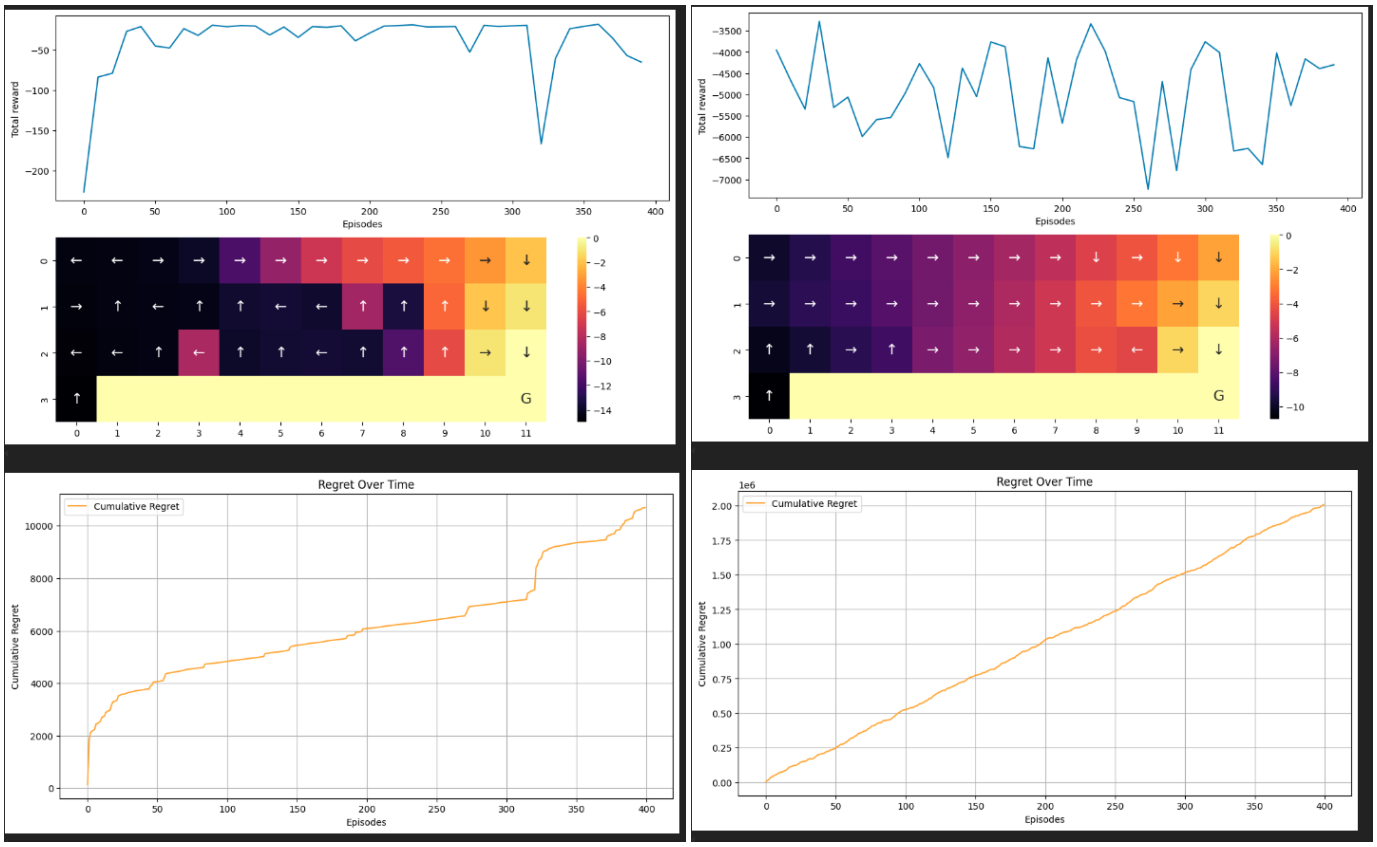
In the learning iterations, we might obtain an intermediate policy that leads to bad trajectories and has high regret. But because of the low epsilon, it might take time to fix the policy by exploration and have a jump in regret.

1.3 Epsilon 0.9 changes linearly. Why? [2.5-points]

Because it almost feels like it's a completely random policy so we have bad trajectories regardless of the Q we learned.

1.4 Compare the policy for epsilon values 0.1 and 0.9. How do they differ, and why do they look different? [2.5-points]

We have epsilon $= 0.1$ on the left and epsilon $= 0.9$ on the right. As we see the epsilon of 0.9 doesn't work at all. It continues to ignore the learnt policy 90% of the time and it has high regret and low reward. But ep=0.1 learns the good policy after $\tilde{3}0$ episodes and stays almost consistent.



1.5 In the epsilon decay section, analyze the optimal policy for the row adjacent to the cliff (the lowest row). Then, compare the different learned policies and their corresponding rewards. [2.5-points]

All the 3 policies with epsilon decay work bad and have all up for the policy for that row. But the slow decay works the worst between the 3 because of the lack of exploitation probably.

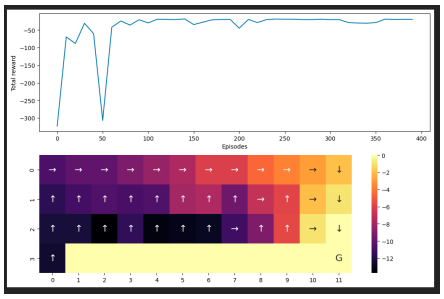
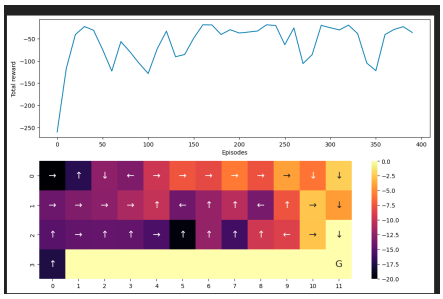
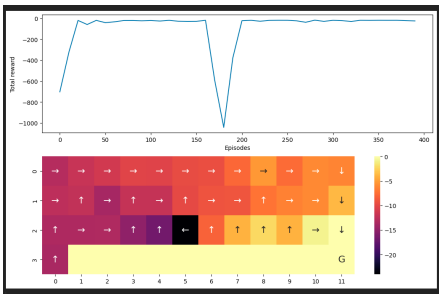
2 N-step Sarsa and N-step Q-learning

2.1 What is the difference between Q-learning and sarsa? [2.5-points]

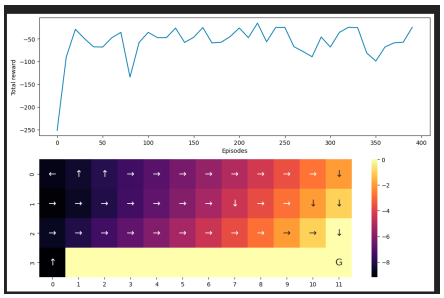
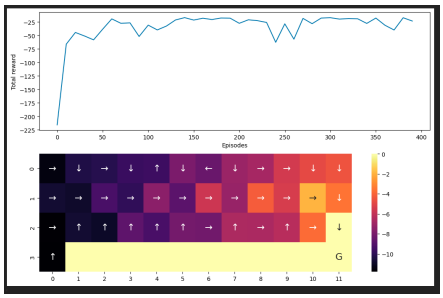
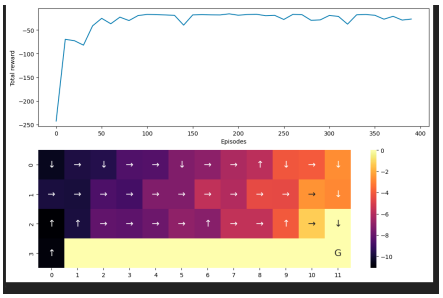
The difference is that sarsa is on-policy and it uses the policy's generated action for the next state to estimate the TD difference, but q-learning uses greedy action for the next state's value to update the TD and therefore is not dependent to the behavior of the policy and is off-policy.

2.2 Compare how different values of n affect each algorithm's performance separately. [2.5-points]

In sarsa, all 3 have the same mean reward of -17 in the evaluation, but n=1 and n=5 seem to have more consistency, the higher the n, the higher the total values for Q are. Which is explainable because it sees a further horizon with higher rewards in the end. Also the higher n should have better policy as a result.

In q-learning, the best mean reward was for n=1, which produced all right actions for the most bottom row. The n=2 agent had the best predicted rewards (≈25). But this is different with our expectation because we expected higher n to produce better agents.

2.3 Is a Higher or Lower n Always Better? Explain the advantages and disadvantages of both low and high n values. [2.5-points]

No, it's not always the same thing. High n values give better learning because of more accurate sampling by looking further in the time. But it results in more delayed updates to the Q-table and also it results in high variance because as n increases, we get closer to MC sampling.

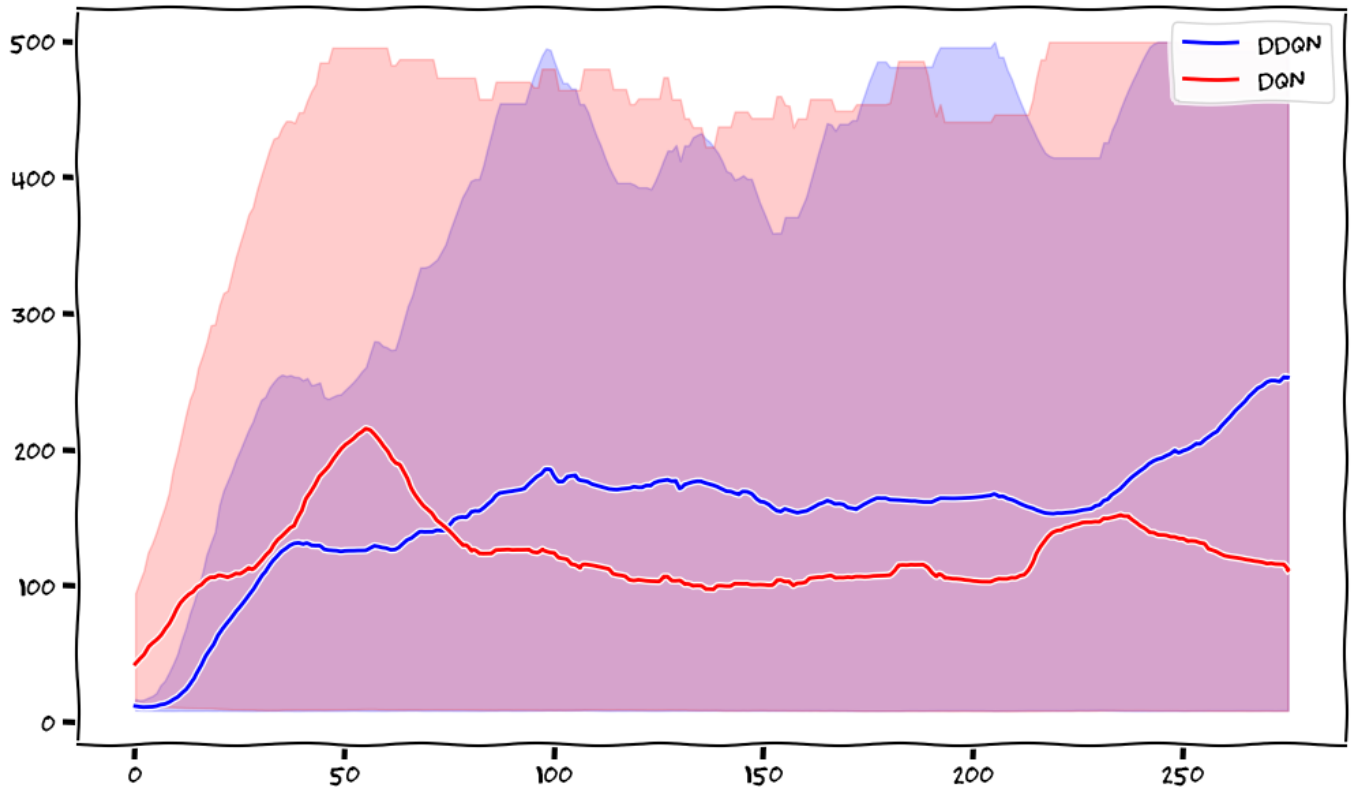
3 DQN vs. DDQN

3.1 Which algorithm performs better and why? [3-points]

The DDQN. It has more stable learning and the target network not moving, the learning happened more smoothly.

3.2 Which algorithm has a tighter upper and lower bound for rewards. [2-points]

Both the algorithms have 0 lower bound because of the bad agents. But the higher bound for the DQN is higher because one agent learned a great policy. But the DDQN have better mean reward with tighter lower-upper bounds.



3.3 Based on your previous answer, can we conclude that this algorithm exhibits greater stability in learning? Explain your reasoning. [2-points]

Yes, it does. We see that the DDQN algorithm starts with less reward initially, which is explainable because two networks are being learned, and it takes more episodes. After the 50th episode, due to DDQN's greater consistency, the average rewards become higher as expected, and more agents have high rewards.

3.4 What are the general issues with DQN? [2-points]

The issue is that the trainable network itself is being used for target value of the loss. And the ongoing changes of the target network causes inefficiency in learning. Also the deep network might estimate wrong q-values in the beginning and it causes the trajectories generated to be in the bad part of the space and never recover.

3.5 How can some of these issues be mitigated? (You may refer to external sources such as research papers and blog posts be sure to cite them properly.) [3-points]

Double DQN helps reduce overestimation bias by separating the processes of selecting and evaluating actions, which leads to more accurate Q-value estimates. Prioritized Experience Replay changes the sampling process by giving more importance to experiences that are likely to have a bigger impact on learning, so the agent can learn faster by focusing on the most informative experiences. Meanwhile, Multi-Step Bootstrapping improves the learning update by incorporating rewards over several future steps instead of just one, which helps balance the bias-variance trade-off and contributes to more stable learning.

3.6 Based on the plotted values in the notebook, can the main purpose of DDQN be observed in the results? [2-points]

Yes it can, as explained earlier, the DDQN takes longer to learn both the target and policy net, but afterwards we can see that it helped the learning phase.

3.7 The DDQN paper states that different environments influence the algorithm in various ways. Explain these characteristics (e.g., complexity, dynamics of the environment) and their impact on DDQN's performance. Then, compare them to the CartPole environment. Does CartPole exhibit these characteristics or not? [4-points]

The DDQN paper points out that the nature of the environment – like its complexity and how dynamic it is – can really affect the algorithm's performance. In more complex environments with tons of states and actions, DDQN can take longer to learn the right policies because there's just so much more for the networks to figure out. Also, in environments where things change in unpredictable or noisy ways, the reward signals get all over the place, which makes it tougher for DDQN to consistently nail down the best actions. So, whether the environment is super complex or has really dynamic, erratic changes, it ends up influencing how fast and stable DDQN learns over time. The cartpole environment is a pretty simple environment and even the simple DQN or DDQN can capture the features and learn the dynamics and the optimal policy consistantly and we are good.

3.8 How do you think DQN can be further improved? (This question is for your own analysis, but you may refer to external sources such as research papers and blog posts be sure to cite them properly.) [2-points]

Maybe one thing is a weighted sampling of the experience replay. Some of the samples which align more closely to the current policy might be more important.

References

[1] R. Sutton and A. Barto, Reinforcement Learning: An Introduction, 2nd Edition, 2020. Available: http://incompleteideas.net/book/the-book-2nd.html.

[2] Gymnasium Documentation. Available: https://gymnasium.farama.org/

[3] Grokking Deep Reinforcement Learning. Available: https://www.manning.com/books/grokking-deep-reinforcement-learning

[4] Deep Reinforcement Learning with Double Q-learning. Available: https://arxiv.org/abs/1509.06461

[5] Cover image designed by freepik